aws

开发人员指南

Amazon CloudFront

Amazon CloudFront: 开发人员指南

Table of Contents

什么是 Amazon CloudFront？

Amazon CloudFront 是一项加快将静态和动态 Web 内容（例如 .html、.css、.js 和图像文件）分发给用户的速度的 Web 服务。CloudFront 通过全球数据中心（称作边缘站点）网络传输内容。当用户请求您用 CloudFront 提供的内容时，请求被路由到提供最低延迟（时间延迟）的边缘站点，从而以尽可能最佳的性能传送内容。

- 如果该内容已经在延迟最短的边缘站点上，CloudFront 将直接提供它。
- 如果内容不在边缘站点中，CloudFront 将从已定义的源（例如，已确定为内容最终版本的来源的 Amazon S3 存储桶、MediaPackage 通道或 HTTP 服务器，如 Web 服务器）检索内容。

例如，假设您要从传统的 Web 服务器中提供图像，而不是从 CloudFront 中提供图像。例如，您可能会使用 URL `https://example.com/sunsetphoto.png` 提供图像 sunsetphoto.png。

您的用户可以轻松导航到该 URL 并查看图像。但他们可能不知道其请求从一个网络路由到另一个网络（通过构成互联网的相互连接的复杂网络集合），直到找到图像。

CloudFront 通过 AWS 主干网络将每个用户请求传送到能以最佳方式提供您的内容的边缘站点，以此来加速分发您的内容。通常，这是向查看器提供传输最快的 CloudFront 边缘服务器。使用 AWS 网络可大大降低用户的请求必须经由的网络数量，从而提高性能。用户将会体验到延迟（加载文件的第一个字节所花费的时间）更短、数据传输速率更高。

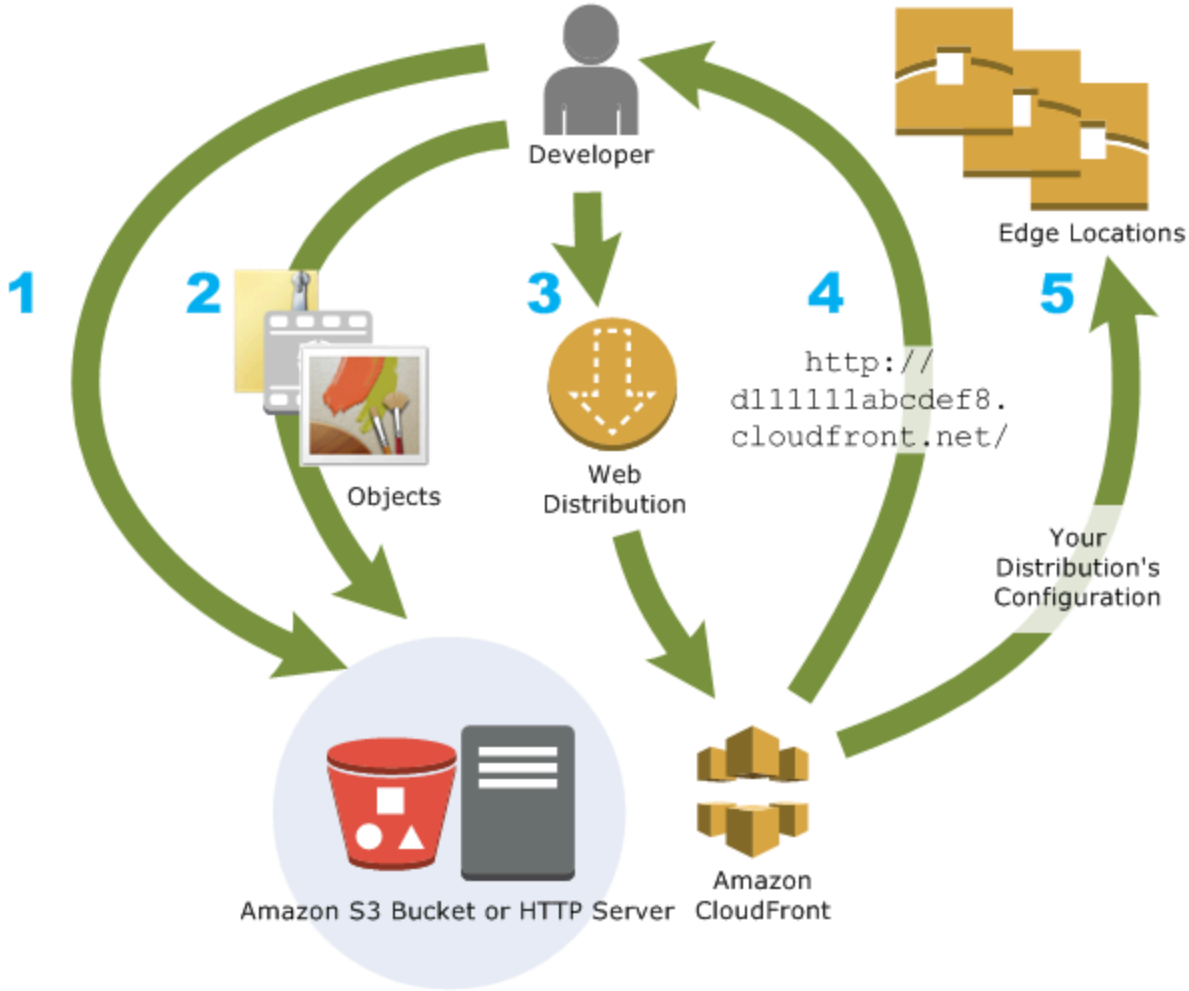
您还会获得更高的可靠性和可用性，因为您的文件（也称为对象）的副本现在存储（或缓存）在全球各地的多个边缘站点上。

主题

- 如何设置 CloudFront 以提供内容
- 定价
- CloudFront 使用方法
- CloudFront 如何交付内容
- CloudFront 边缘服务器的位置和 IP 地址范围
- 将 CloudFront 与 AWS SDK 配合使用
- CloudFront 技术资源

如何设置 CloudFront 以提供内容

您创建 CloudFront 分配，以告知 CloudFront 您希望内容从何处传输，并告知有关如何跟踪和管理内容传输的详细信息。然后，当有人想查看或使用内容时，CloudFront 使用靠近您的查看器的计算机（边缘服务器）快速传输内容。



如何配置 CloudFront 以提供您的内容

1. 您指定源服务器（如 Amazon S3 存储桶或您自己的 HTTP 服务器），CloudFront 将从该服务器获取您的文件，这些文件随后将从全世界的 CloudFront 边缘站点分配。

   源服务器将存储您的对象的原始最终版本。如果您通过 HTTP 提供内容，您的源服务器将为 Amazon S3 存储桶或 HTTP 服务器，例如，Web 服务器。您的 HTTP 服务器可以在 Amazon Elastic Compute Cloud (Amazon EC2) 实例上运行，也可以在您管理的服务器上运行；这些服务器也称为自定义源。

2. 您将您的文件上传至您的源服务器。您的文件也称为对象，通常包括网页、图像和媒体文件，但可以是可通过 HTTP 提供的任何内容。

   如果您将 Amazon S3 存储桶用作源服务器，则可以将存储桶中的对象设为公开可读，这样知道这些对象的 CloudFront URL 的任何人都可以访问它们。您还可以选择将对象设为私有，并控制哪些人可以访问它们。请参阅 使用签名 URL 和签名 Cookie 提供私有内容。

3. 创建一项 CloudFront 分配，此项分配将在用户通过您的网站或应用程序请求文件时告诉 CloudFront 从哪些源服务器获取您的文件。同时，您还需指定一些详细信息，如您是否希望 CloudFront 记录所有请求以及您是否希望此项分配创建后便立即启用。

4. CloudFront 为新分配指定一个域名，您可以在 CloudFront 控制台中查看该域名，该域名也可能被返回以响应编程请求（例如 API 请求）。如果您愿意，您可以添加要改用的备用域名。

5. CloudFront 将您的分配的配置（而不是您的内容）发送到其所有边缘站点或节点 (POP) – 它们是位于地理位置分散的数据中心（CloudFront 在其中缓存您的文件的副本）内的服务器的集合。

您在开发网站或应用程序时，需使用 CloudFront 为您的 URL 提供的域名。例如，如果 CloudFront 返回 `d111111abcdef8.cloudfront.net` 作为您的分配的域名，则 Amazon S3 存储桶中（或 HTTP 服务器上的根目录中）的 logo.jpg 的 URL 将为 `https://d111111abcdef8.cloudfront.net/logo.jpg`。

或者，您可以设置 CloudFront 对您的分配使用您自己的域名。在这种情况下，URL 可能是 `https://www.example.com/logo.jpg`。

（可选）您可配置您的源服务器以向文件添加标头，表示您希望文件在 CloudFront 边缘站点的缓存中保留的时长。默认情况下，每个文件在边缘站点中保留 24 个小时后即会过期。最小过期时间为 0 秒；没有最大过期时间。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

定价

CloudFront 对从其边缘站点传出的数据以及 HTTP 或 HTTPS 请求收费。定价因使用类型、地理区域和功能选择而异。

使用 Amazon Simple Storage Service（Amazon S3）、Elastic Load Balancing 或 Amazon API Gateway 等 AWS 源时，从您的源到 CloudFront 的数据传输始终是免费的。使用 AWS 源，您只需为从 CloudFront 向查看器传输的出站数据付费。

有关更多信息，请参阅 CloudFront 定价以及 Billing and Savings Bundle 常见问题解答。

CloudFront 使用方法

通过使用 CloudFront，可帮助您实现各种目标。此部分列出了其中的几个以及指向更多信息的链接，以便让您了解这些可能性。

主题

加快静态网站内容分发速度

CloudFront 可以加快将静态内容（例如，图像、样式表、JavaScript 等）分发给全球范围内的查看器的速度。通过使用 CloudFront，您可以充分利用 AWS 主干网络和 CloudFront 边缘服务器以在查看器访问您的网站时为其提供快速、安全、可靠的体验。

存储和交付静态内容的简单方式是使用 Amazon S3 存储桶。将 S3 与 CloudFront 结合使用可获得许多好处，包括可以选择使用源访问控制来轻松限制对您的 S3 内容的访问。

有关将 S3 与 CloudFront 结合使用的更多信息（包括帮助您快速入门的 AWS CloudFormation 模板），请参阅 Amazon S3 + Amazon CloudFront：在云中进行的匹配。

提供点播视频或实时流视频

CloudFront 提供了多个选项来将媒体流式传输到全球查看器 – 预先录制的文件和实时内容。

- 对于点播视频 (VOD) 流，您可以使用 CloudFront 以常见格式（如 MPEG DASH、Apple HLS、Microsoft Smooth Streaming 和 CMAF）将内容流式传输到任何设备。
- 对于广播实时流，您可以在边缘站点缓存媒体片段，以便将按正确顺序传输片段的清单文件的多个请求组合起来，从而减小源服务器的负载。

有关如何使用 CloudFront 传输流内容的更多信息，请参阅使用 CloudFront 的点播视频和实时流视频。

在整个系统处理过程中加密特定字段

在对 CloudFront 配置 HTTPS 时，您已获得与源服务器的安全的端到端连接。在添加字段级加密时，您可以在整个系统处理过程中保护特定的数据并实施 HTTPS 安全，以便只有源中的某些应用程序可以查看数据。

要设置字段级加密，您可以将公有密钥添加到 CloudFront，然后指定要使用该密钥加密的字段集。有关更多信息，请参阅 使用字段级加密帮助保护敏感数据。

在边缘进行自定义

通过在边缘站点上运行无服务器代码，开启了为查看器自定义内容和体验的很多可能性，并减少了延迟。例如，您可以在源服务器停机进行维护时返回自定义错误消息，查看器不会获得一般 HTTP 错误消息。或者，您可以在 CloudFront 将请求转发到您的源之前，使用函数来帮助向用户授权并控制对您的内容的访问。

将 Lambda@Edge 与 CloudFront 配合使用，可以通过多种方式自定义 CloudFront 提供的内容。要详细了解 Lambda@Edge 以及如何使用 CloudFront 创建和部署函数，请参阅使用 Lambda@Edge 在边缘进行自定义。要查看可为您自己的解决方案自定义的大量代码示例，请参阅 Lambda@Edge 函数示例。

使用 Lambda@Edge 自定义提供私有内容

除了使用已签名的 URL 或已签名的 Cookie 之外，还可使用 Lambda@Edge 帮助您配置 CloudFront 分配以提供来自您自己的自定义源的私有内容。

要使用 CloudFront 提供私有内容，请执行以下操作：

- 要求用户（查看器）使用已签名的 URL 或已签名的 Cookie 来访问内容。
- 限制对源的访问，以便只能从 CloudFront 的面向源的服务器上进行访问。为此，可以执行以下操作之一：
  - 对于 Amazon S3 源，您可以使用源访问控制（OAC）。
  - 对于自定义源，您可以执行以下操作：
    - 如果自定义源受 Amazon VPC 安全组或 AWS Firewall Manager 保护，您可以使用 CloudFront 托管式前缀列表，以便仅允许从 CloudFront 面向源的 IP 地址传入到源的流量。
    - 使用自定义 HTTP 标头将访问限制为仅来自 CloudFront 的请求。有关更多信息，请参阅the section called “在自定义源上限制对文件的访问”和the section called “向源请求添加自定义

标头”。有关使用自定义标头限制对应用程序负载均衡器源的访问的示例，请参阅the section called “限制访问应用程序负载均衡器”。

- 如果自定义源需要自定义访问控制逻辑，则可以使用 Lambda@Edge 来实现该逻辑，如以下博文中所述：使用 Amazon CloudFront 和 Lambda@Edge 提供私有内容。

CloudFront 如何交付内容

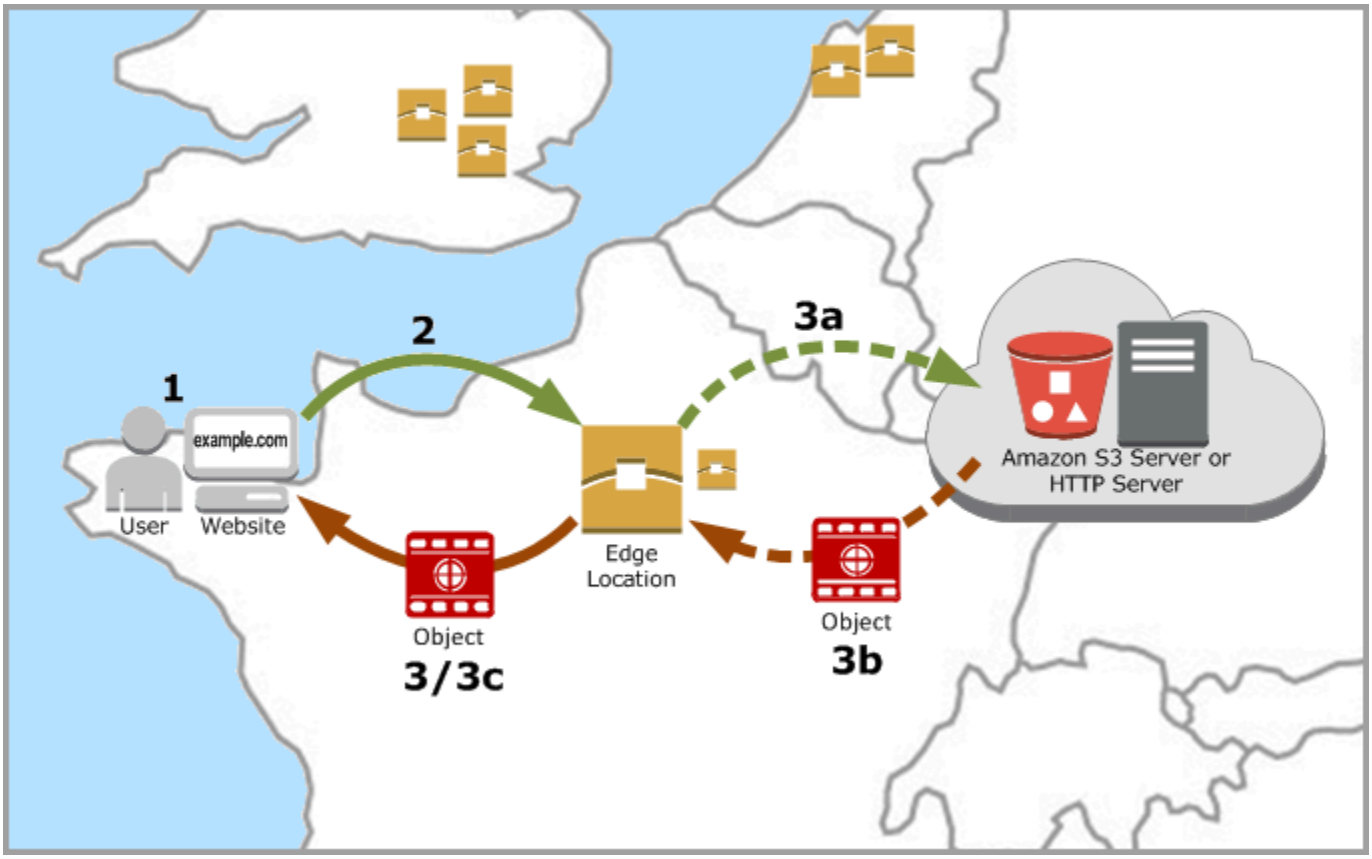
经过一些初步的设置，CloudFront 与您的网站或应用程序协作并加快内容交付速度。此部分说明 CloudFront 如何在查看器请求您的内容时提供该内容。

主题

CloudFront 如何向用户提供内容

配置 CloudFront 以传输您的内容后，用户请求您的对象时将发生以下操作：

1. 用户访问您的网站或应用程序，并发送对于某个对象的请求，例如图像文件和 HTML 文件。
2. DNS 将该请求传送到能以最佳方式满足该请求的 CloudFront POP（边缘站点），通常是以延迟来衡量最近的 CloudFront POP 边缘站点。
3. CloudFront 检查其缓存中是否有所请求的对象。如果对象在缓存中，CloudFront 会将它返回给用户。如果对象不 在缓存中，CloudFront 将执行以下操作：
   a. CloudFront 将该请求和分配中的说明进行比较，然后针对相应的对象将此请求转发到源服务器，例如，转发到 Amazon S3 存储桶或 HTTP 服务器。
   b. 源服务器将此对象发回给边缘站点。
   c. 源中的第一个字节到达后，CloudFront 就开始将此对象转发到用户。CloudFront 还将此对象添加到缓存中，方便下次有人请求该对象。



CloudFront 如何使用区域边缘缓存

CloudFront 节点（也称为 POP 或边缘站点）可确保将受欢迎的内容快速提供给查看器。CloudFront 还具有区域边缘缓存，该缓存可让您的更多内容更靠近查看器（即使该内容的受欢迎程度不足以使其位于 POP 上）以帮助改善该内容的表现。

区域边缘缓存可为所有类型的内容提供帮助，特别是随着时间的推移变得不太常用的内容。这样的示例包括用户生成的内容，例如视频、照片或插图；电子商务资产，例如产品照片和视频；以及新闻和事件相关的内容 (可能突然受到大众欢迎)。

区域缓存的工作方式

区域边缘缓存是全球范围内部署的 CloudFront 位置，靠近查看器。它们位于源服务器和 POP（即全球边缘站点）之间，全球边缘站点直接为查看器提供内容。当对象的受欢迎程度降低时，各个 POP 可能会删除这些对象以便为更受欢迎的内容腾出空间。区域边缘缓存具有比各 POP 更大的缓存，因此对象将在最近的区域边缘缓存位置的缓存中保留更长时间。这有助于让更多内容更为靠近读者，减少 CloudFront 返回源服务器的需要，提升读者阅读体验。

当查看器在您的网站上或通过您的应用程序发出请求时，DNS 将请求传送到能以最佳方式满足用户请求的 POP。就延迟而言，此位置通常是最近的 CloudFront 边缘站点。在 POP 中，CloudFront 检查其

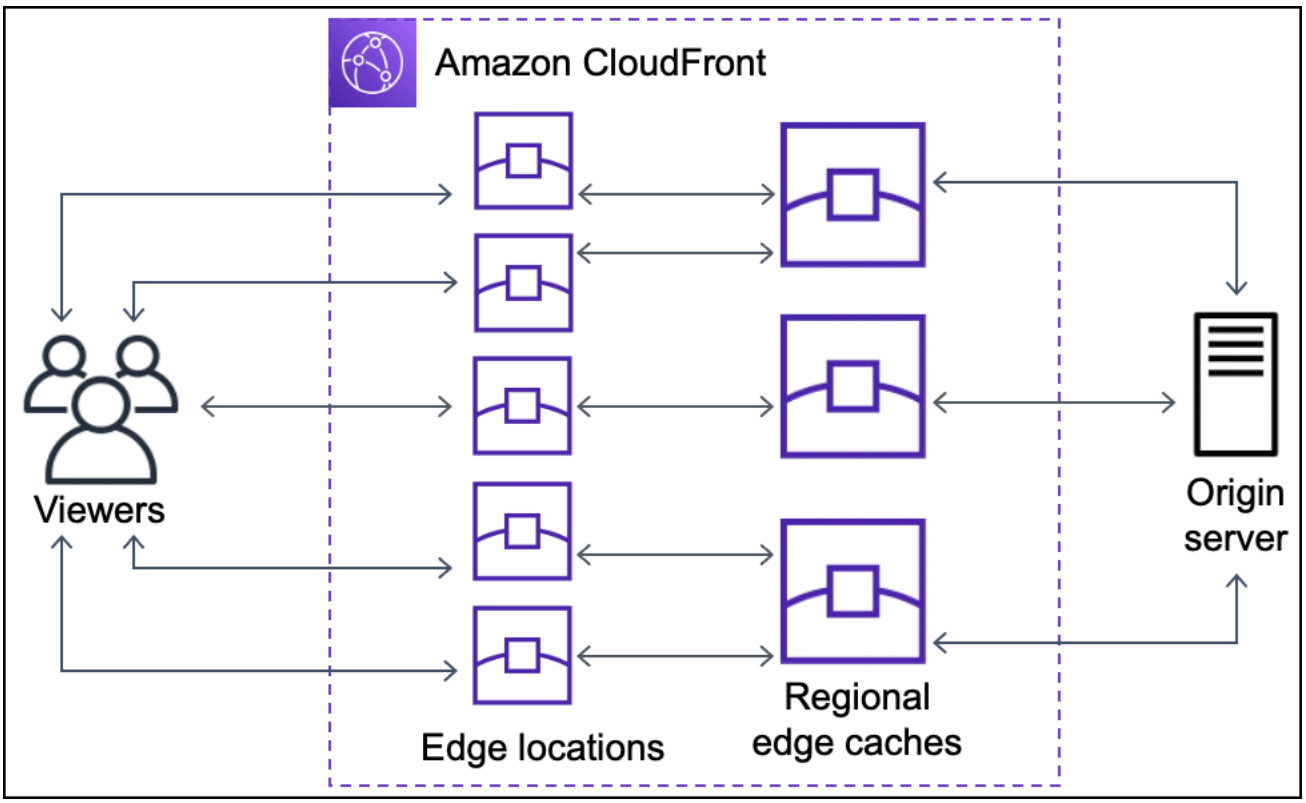
缓存中是否存在所请求的对象。如果对象在缓存中，CloudFront 会将它返回给用户。如果对象不在缓存中，POP 通常将转到最近的区域边缘缓存以提取此对象。有关 POP 何时跳过区域边缘缓存并直接转到源的更多信息，请参阅以下注释。

在区域边缘缓存位置中，CloudFront 将再次检查其缓存中是否有请求的对象。如果对象位于缓存中，则 CloudFront 将其转发到请求它的 POP。区域边缘缓存位置的第一个字节到达后，CloudFront 就开始将对象转发到用户。CloudFront 还将此对象添加到 POP 中的缓存中，方便下次有人请求该对象。

对于未在 POP 和区域边缘缓存位置缓存的对象，CloudFront 将请求与分配中的说明进行比较，并将请求转发到源服务器。在源服务器将对象发送回区域边缘缓存位置后，此对象会转发到 POP，并且 CloudFront 将它转发到用户。在这种情况下，CloudFront 还会将此对象添加到区域边缘缓存位置中的缓存以及 POP，方便查看器下次请求此对象。这将确保区域中的所有 POP 都共享本地缓存，从而消除了针对源服务器的多个请求。CloudFront 还将保留与源服务器的持久性连接，以便尽快从源中提取对象。

> **Note**
>
> - 区域边缘缓存具有与 POP 同等的功能。例如，缓存失效请求将在对象过期之前，同时从 POP 缓存和区域边缘缓存中删除对象。查看器下次请求对象时，CloudFront 将返回源以获取对象的最新版本。
> - 代理 HTTP 方法（`PUT`、`POST`、`PATCH`、`OPTIONS` 和 `DELETE`）直接从 POP 流入源，并且不会通过代理流过区域边缘缓存。
> - 在请求时确定的动态请求不会流经区域边缘缓存，而是直接到达源。
> - 当源是 Amazon S3 存储桶且请求的最佳区域边缘缓存与 S3 存储桶位于同一 AWS 区域时，POP 会跳过区域边缘缓存，直接进入 S3 存储桶。

下图说明了请求和响应如何流经 CloudFront 边缘站点和区域边缘缓存。



CloudFront 边缘服务器的位置和 IP 地址范围

有关 CloudFront 边缘服务器位置的列表，请参阅 Amazon CloudFront 全球边缘网络页面。

Amazon Web Services (AWS) 以 JSON 格式发布其当前的 IP 地址范围。要查看当前范围，请下载 ip-ranges.json。有关更多信息，请参阅《Amazon Web Services 一般参考》中的 AWS IP 地址范围。

要查找与 CloudFront 边缘服务器关联的 IP 地址范围，请在 ip-ranges.json 中搜索以下字符串：

```
"region": "GLOBAL",
"service": "CLOUDFRONT"
```

或者，您可以在 https://d7uri8nf7uskq.cloudfront.net/tools/list-cloudfront-ips 仅查看 CloudFront IP 范围。

使用 CloudFront 托管前缀列表

CloudFront 托管式前缀列表包含所有 CloudFront 全球分布的面向源的服务器的 IP 地址范围。如果源托管在 AWS 上且受 Amazon VPC 安全组保护，则您可以使用 CloudFront 托管式前缀列表仅允许从 CloudFront 面向源的服务器传入到源的流量，从而阻止任何非 CloudFront 流量达到您的

源。CloudFront 使用所有 CloudFront 全球面向源的服务器的 IP 地址来维护托管式前缀列表，使其始终保持最新状态。使用 CloudFront 托管式前缀列表，您无需自己读取或维护 IP 地址范围列表。

例如，假设您的源是位于欧洲地区（伦敦）区域（`eu-west-2`）的 Amazon EC2 实例。如果实例位于 VPC 中，则可以创建安全组规则，允许从 CloudFront 托管前缀列表的入站 HTTPS 访问。这使所有 CloudFront 全球面向源的服务器均可达到此实例。如果您从安全组中删除所有其他入站规则，则可以阻止任何非 CloudFront 流量到达该实例。

CloudFront 托管前缀列表被命名为 com.amazonaws.global.cloudfront.origin-facing。有关更多信息，请参阅《Amazon VPC 用户指南》中的 使用 AWS 托管前缀列表。

> **Important**
>
> CloudFront 托管前缀列表在其应用于 Amazon VPC 配额的方式方面是唯一的。有关更多信息，请参阅《Amazon VPC 用户指南》中的AWS 托管前缀列表权重。

将 CloudFront 与 AWS SDK 配合使用

AWS 软件开发工具包 (SDK) 适用于许多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档	代码示例
AWS SDK for C++	AWS SDK for C++ 代码示例
AWS CLI	AWS CLI 代码示例
AWS SDK for Go	AWS SDK for Go 代码示例
AWS SDK for Java	AWS SDK for Java 代码示例
AWS SDK for JavaScript	AWS SDK for JavaScript 代码示例
AWS SDK for Kotlin	AWS SDK for Kotlin 代码示例
AWS SDK for .NET	AWS SDK for .NET 代码示例
AWS SDK for PHP	AWS SDK for PHP 代码示例

SDK 文档	代码示例
AWS Tools for PowerShell	Tools for PowerShell 代码示例
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 代码示例
AWS SDK for Ruby	AWS SDK for Ruby 代码示例
AWS SDK for Rust	AWS SDK for Rust 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
AWS SDK for Swift	AWS SDK for Swift 代码示例

示例可用性

找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

CloudFront 技术资源

使用以下资源获取有关 CloudFront 技术问题的答案：

- AWS re:Post – 基于社区的问答网站，供开发人员讨论与 CloudFront 相关的技术问题。
- AWS Support 中心 – 该网站包含有关您最近的支持案例和 AWS Trusted Advisor 返回的结果以及运行状况检查的信息。它还提供指向开发论坛、技术常见问题解答、服务运行状况控制面板以及 AWS Support 计划相关信息的链接。
- AWS Premium Support – 了解有关 AWS Premium Support 的信息，该服务是一种一对一的快速响应支持渠道，可帮助您在 AWS 上构建和运行应用程序。
- AWS IQ – 获得来自 AWS 认证的专业人员和专家的帮助。

CloudFront 入门

此部分中的主题向您展示如何开始通过 Amazon CloudFront 交付内容。

设置主题介绍了以下教程的先决条件，例如创建AWS 账户和创建具有管理权限的用户。

基础分配教程将向您演示如何设置源访问控制（OAC，Origin Access Control），以便将经过身份验证的请求发送到 Amazon S3 源。

安全静态网站教程向您演示如使用 OAC 和 Amazon S3 源，为您的域名创建安全静态网站。本教程使用 Amazon CloudFront（CloudFront）模板进行配置和部署。

主题

设置

本主题介绍了准备使用 Amazon CloudFront 的预备步骤（如创建 AWS 账户）。

主题

注册 AWS 账户

如果您还没有 AWS 账户，请完成以下步骤来创建一个。

注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

当您注册 AWS 账户时，系统将会创建一个 AWS 账户根用户。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

注册过程完成后，AWS 会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册 AWS 账户 后，请保护好您的 AWS 账户根用户，启用 AWS IAM Identity Center，并创建一个管理用户，以避免使用根用户执行日常任务。

保护您的 AWS 账户根用户

1. 选择根用户并输入您的 AWS 账户电子邮件地址，以账户拥有者身份登录 AWS Management Console。在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅《IAM 用户指南》中的为 AWS 账户根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关如何使用 IAM Identity Center 目录 作为身份源的教程，请参阅《AWS IAM Identity Center 用户指南》中的使用默认的 IAM Identity Center 目录 配置用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

要获取使用 IAM Identity Center 用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

选择如何访问 CloudFront

您可以通过以下方式访问 Amazon CloudFront：

- AWS Management Console – 该指南中的过程介绍了如何使用 AWS Management Console 执行任务。
- AWS 开发工具包 – 如果您使用 AWS 为其提供开发工具包的编程语言，您可以使用开发工具包访问 CloudFront。开发工具包可简化身份验证、与您的开发环境轻松集成，并有助于访问 CloudFront 命令。有关更多信息，请参阅 将 CloudFront 与 AWS SDK 配合使用。
- CloudFront API – 如果要使用开发工具包不可用的编程语言，请参阅《Amazon CloudFront API 参考》，以了解有关 API 操作及如何发出 API 请求的信息。
- AWS CLI – AWS Command Line Interface（AWS CLI）是一个用于管理AWS 服务的统一工具。有关如何安装和配置 AWS CLI 的信息，请参阅《AWS Command Line Interface 用户指南》中的安装或更新 AWS CLI 的最新版本。
- 适用于 Windows PowerShell 的工具 – 如果您有使用 Windows PowerShell 的经验，则可能更愿意使用 AWS Tools for Windows PowerShell。有关更多信息，请参阅 AWS Tools for Windows PowerShell 用户指南 中的安装 AWS Tools for Windows PowerShell。

开始使用基本 CloudFront 分配

此部分中的过程将介绍如何使用 CloudFront 设置执行以下操作的基本配置：

- 创建用作分配源的存储桶。

- 将对象的原始版本存储在一个 Amazon Simple Storage Service (Amazon S3) 存储桶中
- 使用源访问控制（OAC）将经过身份验证的请求发送到您的 Amazon S3 源。OAC 通过 CloudFront 发送请求，以防止查看器直接访问您的 S3 存储桶。有关 OAC 的更多信息，请参阅限制对 Amazon Simple Storage Service 源的访问。
- 将 URL 中的 CloudFront 域名用于您的对象（例如，`https://d111111abcdef8.cloudfront.net/index.html`）
- 在 CloudFront 边缘站点上将您的对象保留默认的 24 小时持续时间（最短持续时间为 0 秒）

其中的大多数选项是可自定义的。有关如何自定义 CloudFront 分配选项的信息，请参阅创建分配。

主题

先决条件

开始之前，请确保您已完成设置中的步骤。

步骤 1：创建 Amazon S3 存储桶

Amazon S3 存储桶是文件（对象）或文件夹的容器。当 Amazon S3 存储桶作为源时，CloudFront 可以为您分发几乎任何类型的文件。例如，CloudFront 可以分配文本、图像和视频。您可以在 Amazon S3 上存储的数据量没有上限。

在本教程中，您将使用提供的 `hello world` 文件示例创建一个 S3 存储桶，您将使用这些文件创建基本网页。

创建桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。

2. 建议您使用我们的 Hello World 示例作为入门指导。下载 hello world 网页：hello-world-html.zip。将其解压缩，然后将 `css` 文件夹和 `index` 文件保存到方便的位置，例如运行浏览器的桌面。
3. 选择创建存储桶。
4. 输入符合《Amazon Simple Storage Service 用户指南》中通用存储桶命名规则的唯一存储桶名称。
5. 对于区域，建议选择地理位置靠近您的 AWS 区域。（这可以降低延迟和成本。）
   - 也可以选择其他区域。例如，您可以这样做来满足监管要求。
6. 将所有其他设置保留默认值，然后选择创建存储桶。

步骤 2：将内容上传到桶

创建 Amazon S3 存储桶后，将解压缩的 `hello world` 文件内容上传到该存储桶。（您已在 步骤 1：创建 Amazon S3 存储桶 中下载并解压缩此文件。）

将内容上传到 Amazon S3

1. 在通用存储桶部分，选择新存储桶的名称。
2. 选择上传。
3. 在上传页面，将 `css` 文件夹和 `index` 文件拖到拖放区域。
4. 将所有其他设置保留为原定设置值，然后选择上传。

步骤 3：创建使用 Amazon S3 源和 OAC 的 CloudFront 分配

在本教程中，您将创建一个使用 Amazon S3 源和源访问控制（OAC）的 CloudFront 分配。OAC 可帮助您安全地将经过身份验证的请求发送到 Amazon S3 源。有关 OAC 的更多信息，请参阅限制对 Amazon Simple Storage Service 源的访问。

创建具有使用 OAC 的 Amazon S3 源的 CloudFront 分配

1. 通过 打开 CloudFront 控制台https://console.aws.amazon.com/cloudfront/v4/home
2. 选择 Create distribution（创建分配）。
3. 对于源，请在源域中，选择您为本教程创建的 S3 存储桶。
4. 对于源，请在源访问中，选择源访问控制设置（推荐）。
5. 对于源访问控制，请选择创建新 OAC。

6. 在创建新 OAC 窗格中，保留默认设置并选择创建。
7. 对于Web 应用程序防火墙（WAF），请选择其中一个选项。
8. 对于所有其他部分和设置，接受默认值。有关这些选项的详细信息，请参阅 分配设置。
9. 选择创建分配。
10. 在需要更新 S3 存储桶策略横幅中，阅读消息并选择复制策略。
11. 在同一横幅中，选择转到 S3 存储桶权限以更新策略链接。（这将转至 Amazon S3 控制台中的存储桶详细信息页面）。
12. 在 Bucket policy（存储桶策略）下，选择 Edit（编辑）。
13. 在编辑语句字段中，粘贴您在步骤 10 中复制的策略。
14. 选择保存更改。
15. 返回 CloudFront 控制台并查看您的新分配的详细信息部分。部署分配后，上次修改时间字段将从正在部署更改为部署日期和时间。
16. 记录 CloudFront 指派给分配的域名。其内容类似于以下所示：`d111111abcdef8.cloudfront.net`。

在生产环境中使用本教程中的分配和 S3 存储桶之前，请务必对其进行配置以满足您的特定需求。有关在生产环境中配置访问权限的信息，请参阅配置安全访问和限制对内容的访问。

步骤 4：通过 CloudFront 访问您的内容

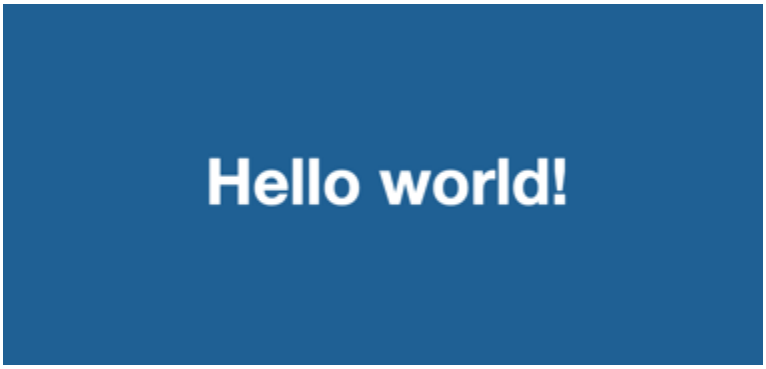
要通过 CloudFront 访问您的内容，请将 CloudFront 分配的域名与内容的主页组合在一起。（您在 步骤 3：创建使用 Amazon S3 源和 OAC 的 CloudFront 分配 中记录了您的分配域名。）

- 您的分配域名可能如下所示：`d111111abcdef8.cloudfront.net`。
- 网站主页的路径通常为 `/index.html`。

因此，通过 CloudFront 访问您的内容的 URL 可能如下所示：

`https://d111111abcdef8.cloudfront.net/index.html`.

如果您按照上述步骤操作并使用了 hello world 网页，您应该看到以下内容：



将更多内容上传到此 S3 桶时，您可以将 CloudFront 分配域名与 S3 桶中的对象路径组合在一起，从而通过 CloudFront 访问内容。例如，如果您将一个名为 `new-page.html` 的新文件上传到 S3 桶的根目录，URL 如下所示：

`https://d111111abcdef8.cloudfront.net/new-page.html`.

第 5 步：清理

如果您仅出于练习目的创建分配和 S3 存储桶，请删除它们，这样就不会再产生费用。首先删除分配。有关更多信息，请参阅以下链接：

- 删除分配
- 删除存储桶

增强您的 CloudFront 基本分配

本入门教程提供创建分配所需的最小框架。建议您探索以下增强功能：

- 默认情况下，Amazon S3 存储桶中的文件（对象）设置为私有的。只有创建了该存储桶的 AWS 账户才有权读取或写入文件。如果要允许任何人使用 CloudFront URL 访问您的 Amazon S3 存储桶中的对象，您必须授予对象公共读取权限。
- 您可以使用 CloudFront 私有内容功能来限制对 Amazon S3 存储桶中内容的访问权限。更多有关分配私有内容的信息，请参阅 使用签名 URL 和签名 Cookie 提供私有内容。
- 您可以将 CloudFront 分配配置为使用自定义域名（例如，使用 `www.example.com` 而不是 `d111111abcdef8.cloudfront.net`）。有关更多信息，请参阅 使用自定义 URL。
- 本教程使用具有源访问控制（OAC）的 Amazon S3 源。但是，如果您的源是配置为网站端点的 S3 存储桶，则无法使用 OAC。在这种情况下，必须通过 CloudFront 将存储桶设置为自定义源。有关更多信息，请参阅 使用配置为网站端点的 Amazon S3 存储桶。有关 OAC 的更多信息，请参阅限制对 Amazon Simple Storage Service 源的访问。

安全静态网站入门

您可以通过使用本主题中介绍的解决方案为您的域名创建安全静态网站来开始使用 Amazon CloudFront。静态网站 仅使用静态文件（如 HTML、CSS、JavaScript、图像和视频），并且不需要服务器或服务器端处理。使用此解决方案，您的网站将获得以下好处：

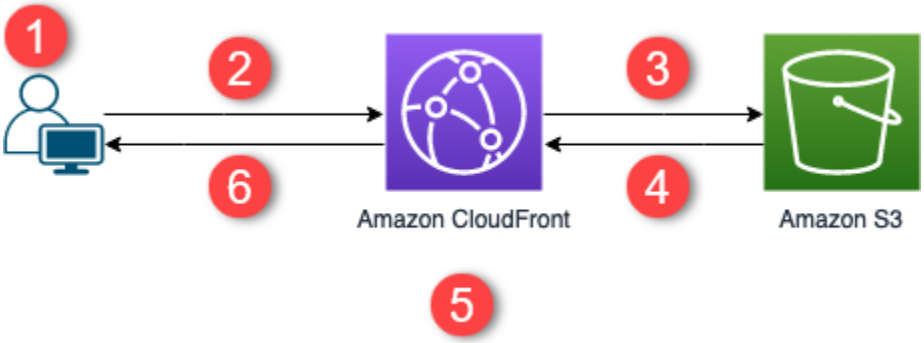
- 使用 Amazon Simple Storage Service (Amazon S3) 的持久存储 – 此解决方案创建 Amazon S3 存储桶来托管静态网站的内容。要更新您的网站，只需将新文件上传到 S3 存储桶。
- 通过 Amazon CloudFront 内容分发网络加快速度 – 此解决方案创建一个 CloudFront 分配，以便以低延迟将您的网站提供给查看器。此分配配置了源访问控制（OAC）功能，以确保只能通过 CloudFront（而不是直接从 S3）访问网站。
- 由 HTTPS 和安全标头提供保护 – 此解决方案在 AWS Certificate Manager（ACM）中创建 SSL/TLS 证书，并将其附加到 CloudFront 分配。此证书使分配能够使用 HTTPS 安全地为您的域名网站提供服务。
- 使用 AWS CloudFormation 进行配置和部署 – 此解决方案使用 AWS CloudFormation 模板来设置所有组件，因此，您可以更多地关注网站的内容，而更少地关注配置组件。

这个解决方案在 GitHub 上是开源的。要查看代码、提交拉取请求或开设问题，请转到 https://github.com/aws-samples/amazon-cloudfront-secure-static-site。

主题

解决方案概述

下图显示此静态网站解决方案的工作原理的概述：



1. 查看器在 www.example.com 上请求访问网站。
2. 如果缓存请求的对象，则 CloudFront 将对象从其缓存返回给查看器。
3. 如果对象不在 CloudFront 缓存中，CloudFront 会从源（S3 存储桶）请求对象。
4. S3 将对象返回到 CloudFront。
5. CloudFront 会缓存该对象。
6. 该对象将返回到查看器。对去往同一 CloudFront 边缘站点的对象的后续请求将从 CloudFront 缓存中提供服务。

部署解决方案

要部署此安全静态网站解决方案，您可以从以下任一选项中进行选择：

- 使用 AWS CloudFormation 控制台部署具有默认内容的解决方案，然后将您的网站内容上传到 Amazon S3。
- 将解决方案克隆到您的计算机以添加您的网站内容。然后，使用 AWS Command Line Interface (AWS CLI) 部署解决方案。

> **Note**
>
> 您必须使用美国东部（弗吉尼亚州北部）区域来部署 CloudFormation 模板。

主题

先决条件

要使用此解决方案，您必须具有以下先决条件：

- 指向 Amazon Route 53 托管区域的注册域名（例如 example.com）。托管区域必须位于您部署此解决方案的同一 AWS 账户中。如果您没有已注册的域名，可以向 Route 53 注册一个。如果您有注册的域名，但它未指向 Route 53 托管区域，请将 Route 53 配置为您的 DNS 服务。

- AWS Identity and Access Management (IAM) 启动创建 IAM 角色的 CloudFormation 模板的权限，以及在解决方案中创建所有 AWS 资源的权限。

使用此解决方案时产生的费用由您负责。有关成本的更多信息，请参阅每项 AWS 服务的定价页面。

使用 AWS CloudFormation 控制台

使用 CloudFormation 控制台进行部署

1. 选择在 AWS 上启用以在 AWS CloudFormation 控制台中打开此解决方案。如有必要，请登录您的 AWS 账户。

2. 将在 CloudFormation 控制台中打开创建堆栈向导，其中包含指定此解决方案的 CloudFormation 模板的预填充字段。

   在页面底部，选择 Next。

3. 在指定堆栈详细信息页面上，输入以下字段的值：

   - 子域名 – 输入要用于您的网站的子域名。例如，如果子域名为 www，则您的网站在 *www*.example.com 上可用。（将 example.com 替换为您的域名，如以下项目符号中所述。）
   - 域名 – 输入您的域名，如 *example.com*。此域必须指向 Route 53 托管区域。
   - HostedZoneId – 您的域名的 Route 53 托管区域 ID。
   - CreateApex –（可选）在您的 CloudFront 配置中为域 apex（example.com）创建别名。

4. 在完成后，选择下一步。
5. （可选）在配置堆栈选项页面上，添加标签和其他堆栈选项。
6. 在完成后，选择下一步。
7. 在审核页面上，滚动到页面底部，然后选择功能部分中的两个框。这些功能允许 CloudFormation 创建允许访问堆栈资源并动态命名这些资源的 IAM 角色。
8. 选择创建堆栈。
9. 等待堆栈完成创建。堆栈会创建一些嵌套堆栈，并且可能需要几分钟才能完成。完成后，状态将更改为 CREATE_COMPLETE。

   当状态为 CREATE_COMPLETE 时，请转到 https://*www.example.com*，以查看您的网站（将 www.example.com 替换为您在步骤 3 中指定的子域名和域名）。您应该看到网站的默认内容：

将网站的默认内容替换为您自己的内容

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择名称以 amazon-cloudfront-secure-static-site-s3bucketroot- 开头的存储桶。

> **Note**
>
> 请确保选择名称中包含 s3bucketroot 而不是 s3bucketlogs 的存储桶。名称中包含 s3bucketroot 的存储桶包含网站内容。具有 s3bucketlogs 的存储桶只包含日志文件。

3. 删除网站的默认内容，然后上传您自己的内容。

> **Note**
>
> 如果您查看了您的网站以及此解决方案的默认内容，那么某些默认内容可能会缓存在 CloudFront 边缘站点中。要确保查看器看到更新后的网站内容，请使文件无效，以便从 CloudFront 边缘站点删除缓存的副本。有关更多信息，请参阅 使文件失效以删除内容。

本地克隆解决方案

先决条件

要在部署此解决方案之前添加网站内容，您必须在本地打包解决方案的构件，这需要 Node.js 和 npm。有关更多信息，请参阅 https://www.npmjs.com/get-npm。

添加网站内容并部署解决方案

1. 从 克隆或下载解决方案https://github.com/aws-samples/amazon-cloudfront-secure-static-site 克隆或下载后，打开命令提示符或终端并导航到 `amazon-cloudfront-secure-static-site` 文件夹。
2. 运行以下命令来安装并打包解决方案的构件：

```
make package-static
```

3. 将网站的内容复制到 www 文件夹中，覆盖默认网站内容。
4. 运行以下 AWS CLI 命令创建 Amazon S3 存储桶以存储解决方案的构件。用您自己的存储桶名称替换 *example-bucket-for-artifacts*。

```
aws s3 mb s3://example-bucket-for-artifacts --region us-east-1
```

5. 运行以下 AWS CLI 命令，将解决方案构件打包为 CloudFormation 模板。将 *example-bucket-for-artifacts* 替换为您在上一步中创建的存储桶的名称。

```
aws cloudformation package \
    --region us-east-1 \
    --template-file templates/main.yaml \
    --s3-bucket example-bucket-for-artifacts \
    --output-template-file packaged.template
```

6. 运行以下命令，使用 CloudFormation 部署解决方案，同时替换以下值：

   - *your-CloudFormation-stack-name* – 替换为 CloudFormation 堆栈的名称。
   - *example.com* – 替换为您的域名。此域必须指向同一 AWS 账户中的 Route 53 托管区。
   - *www* – 替换为要用于您的网站的子域名。例如，如果子域名为 www，则您的网站在 www.example.com 上可用。
   - *hosted-zone-ID* – 替换为您的域名的 Route 53 托管区 ID。

```
aws cloudformation deploy \
    --region us-east-1 \
    --stack-name your-CloudFormation-stack-name \
    --template-file packaged.template \
    --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \
    --parameter-overrides DomainName=example.com SubDomain=www HostedZoneId=hosted-
zone-ID
```

   - （可选）要使用域 Apex 部署堆栈，请改为运行以下命令。

```
aws --region us-east-1 cloudformation deploy \
    --stack-name your-CloudFormation-stack-name \
    --template-file packaged.template \
```

```
    --capabilities CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \
    --parameter-overrides  DomainName=example.com SubDomain=www
HostedZoneId=hosted-zone-ID CreateApex=yes
```

7. 等待 CloudFormation 堆栈创建完成。堆栈会创建一些嵌套堆栈，并且可能需要几分钟才能完成。完成后，状态将更改为 CREATE_COMPLETE。

   当状态更改为 CREATE_COMPLETE 时，请转到 https://www.example.com 以查看您的网站（将 www.example.com 替换为您在前面步骤中指定的子域名和域名）。您应该看到您网站的内容。

查找访问日志

此解决方案为 CloudFront 分配启用访问日志。完成以下步骤以查找此分配的访问日志。

查找分配的访问日志

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择名称以 amazon-cloudfront-secure-static-site-s3bucketlogs- 开头的存储桶。

> **Note**
>
> 请确保选择名称中包含 s3bucketlogs 而不是 s3bucketroot 的存储桶。名称中具有 s3bucketlogs 的存储桶包含日志文件。带有 s3bucketroot 的存储桶包含网站内容。

3. 名为 cdn 的文件夹包含 CloudFront 访问日志。

配置分配

您需要创建 Amazon CloudFront 分配，以告知 CloudFront 您希望内容从何处传输，并告知有关如何跟踪和管理内容传输的详细信息。

从以下配置设置中进行选择：

- 您的内容源 - CloudFront 从中获取要分配的文件的 Amazon S3 存储桶、AWS Elemental MediaPackage 通道、AWS Elemental MediaStore 容器、Elastic Load Balancing 负载均衡器或 HTTP 服务器。您可以为单个分配指定多达 25 个源的任意组合。
- 访问 – 您是希望所有人都可以访问这些文件，还是仅限于某些用户。
- 安全 – 您是否希望启用 AWS WAF 保护并要求用户使用 HTTPS 访问您的内容。
- 缓存键 – 要包含在缓存键 中的值（如果有）。缓存键唯一标识给定分配的缓存中的每个文件。
- 源请求设置 – 您是否希望 CloudFront 在它发送到源的请求中包含 HTTP 标头、Cookie 或查询字符串。
- 地理限制 – 您是否希望 CloudFront 阻止选定国家/地区的用户访问您的内容。
- 日志 – 您希望 CloudFront 创建标准日志，还是创建显示查看器活动的实时日志。

有关更多信息，请参阅 分配设置参考。

有关您可以为每个 AWS 账户创建的当前最大分配数，请参阅 分配的一般配额。对于您可按照分配提供的文件数没有最大值。

您可以使用分配，通过 HTTP 或 HTTPS 提供以下内容：

- 使用 HTTP 或 HTTPS 的静态和动态下载内容，例如 HTML、CSS、JavaScript 和图像文件。
- 不同格式的点播视频，例如 Apple HTTP Live Streaming（HLS）和 Microsoft Smooth Streaming。有关更多信息，请参阅 通过 CloudFront 提供点播视频。
- 一个实时事件，例如实时会议或音乐会。对于实时流，您可以使用 AWS CloudFormation 堆栈自动创建分配。有关更多信息，请参阅 使用 CloudFront 和 AWS Media Services 提供实时流视频。

以下主题详细介绍了 CloudFront 分配以及如何配置这些分配以满足业务需求。有关创建分配的更多信息，请参阅创建分配。

主题

- 创建分配

- 分配设置参考
- 测试分配
- 更新分配
- 标记分配
- 删除分配
- 使用 CloudFront 持续部署来安全地测试 CDN 配置更改
- 在 CloudFront 分配中使用各种源
- 通过添加备用域名（CNAME）使用自定义 URL
- 将 WebSocket 与 CloudFront 分配结合使用

创建分配

本主题介绍如何使用 CloudFront 控制台创建分配。

创建分配概述

1. 根据您的源服务器创建一个或多个 Amazon S3 存储桶或配置 HTTP 服务器。源是您存储内容的原始版本的位置。当 CloudFront 获得您的文件请求时，它将转到源，以获取其在边缘站点分配的文件。您可使用 Amazon S3 存储桶和 HTTP 服务器的任意组合作为您的源服务器。

   - 如果您使用 Amazon S3，则存储桶的名称必须全部小写，并且不能包含空格。
   - 如果您使用 Amazon EC2 服务器或其他自定义源，请查看使用 Amazon EC2（或其他自定义源）。
   - 有关您可以为分配创建的源的当前最大数量或要请求提高限额，请参阅分配的一般配额。
2. 将内容上传到源服务器。您可以使对象公开可读，也可以使用 CloudFront 签名的 URL 来限制对内容的访问。

   > **Important**
   >
   > 您负责确保源服务器的安全。您必须确保 CloudFront 有权限访问服务器，并确保安全设置可保护您的内容。

3. 创建 CloudFront 分配：

   - 有关在 CloudFront 控制台中创建分配的详细步骤，请参阅创建分配。

- 有关使用 CloudFront API 创建分配的信息，请参阅《Amazon CloudFront API 参考》中的 CreateDistribution。

4. （可选）如果您使用 CloudFront 控制台创建分配，则可为分配创建更多缓存行为或源。有关行为和源的更多信息，请参阅更新 CloudFront 分配。
5. 测试您的分配。有关测试的更多信息，请参阅 测试分配。
6. 开发您的网站或应用程序，以使用 CloudFront 在您在第 3 步创建分配后返回的域名访问您的内容。例如，如果 CloudFront 返回 d111111abcdef8.cloudfront.net 作为分配的域名，则 Amazon S3 存储桶中或 HTTP 服务器上根目录中的文件 `image.jpg` 的 URL 为 `https://d111111abcdef8.cloudfront.net/image.jpg`。

   如果您在创建分配时指定了一个或多个备用域名 (CNAME)，则可使用您自己的域名。在这种情况下，`image.jpg` 的 URL 可能是 `https://www.example.com/image.jpg`。

   请注意以下几点：

   - 如果您想使用签署的 URL 来限制对内容的访问，请参阅 使用签名 URL 和签名 Cookie 提供私有内容。
   - 如果您想提供压缩的内容，请参阅 提供压缩文件。
   - 有关 CloudFront 请求和 Amazon S3 的响应行为以及自定义源的信息，请参阅请求和响应行为。

主题

在控制台中创建 CloudFront 分配

创建分配（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择分配，然后选择创建分配。
3. 指定分配的设置。有关更多信息，请参阅 分配设置参考。
4. 保存您的更改。

5. 在 CloudFront 创建分配后，分配的状态列的值将从正在部署更改为部署分配的日期和时间。如果您选择了启用该分配，那么此时它将准备好处理请求。

   CloudFront 指派给分配的域名将出现在分配列表中。（它同时也出现在选定分发的常规选项卡上。）

   > **Tip**
   >
   > 除了 CloudFront 分配给您的域名，您也可以遵循 通过添加备用域名（CNAME）使用自定义 URL 中的步骤使用替代域名。

6. 当已部署您的分配时，请确认您可使用新的 CloudFront URL 或 CNAME 访问您的内容。有关更多信息，请参阅 测试分配。

CloudFront 在控制台中显示的值

当您创建新分配或更新现有分配时，CloudFront 将在 CloudFront 控制台中显示以下信息。

> **Note**
>
> 有效的可信签署人，具有有效 CloudFront 密钥对并可以用于创建有效签名 URL 的 AWS 账户，目前在 CloudFront 控制台中不可见。

分配 ID

当您使用 CloudFront API 对分配执行操作时，可使用分配 ID 指定要使用的分配，例如 `EDFDVBD6EXAMPLE`。您不能更改分配的分配 ID。

部署和状态

部署分配时，您会在上次修改时间列下看到正在部署状态。等待分配完成部署，并确保状态列显示已启用。有关更多信息，请参阅 分配状态。

上次修改时间

分配上一次修改的日期和时间，使用 ISO 8601 格式，例如，2012-05-19T19:37:58Z。有关更多信息，请参阅 https://www.w3.org/TR/NOTE-datetime。

域名

在对象的链接中使用分配的域名。例如，如果分配的域名为 `d111111abcdef8.cloudfront.net`，`/images/image.jpg` 的链接将为 `https://d111111abcdef8.cloudfront.net/images/image.jpg`。您不能更改分配的 CloudFront 域名。有关对象链接的 CloudFront URL 的更多信息，请参阅在 CloudFront 中自定义文件的 URL 格式。

如果您指定一个或多个备用域名 (CNAME)，则可使用自己的域名，以链接到您的对象，而不是使用 CloudFront 域名。有关别名记录 (CNAME) 的更多信息，请参阅 备用域名 (CNAME)。

> **Note**
>
> CloudFront 域名是唯一的。分配的域名从来不用于先前的分配，并且在将来也绝不会重新用于其他的分配。

其他链接

有关创建分配的更多信息，请参阅以下链接。

- 要了解如何创建使用 Amazon Simple Storage Service（Amazon S3）存储桶源和源访问控制（OAC）的分配，请参阅开始使用基本 CloudFront 分配。
- 有关使用 CloudFront API 创建分配的信息，请参阅《Amazon CloudFront API 参考》中的 CreateDistribution。
- 有关更新分配（例如添加或更改缓存行为）的信息，请参阅更新分配。
- 要查看您可以为每个 AWS 账户创建的当前最大源数量或要请求提高配额（以前称为限制），请参阅分配的一般配额。

分配设置参考

当您使用 CloudFront 控制台创建新的分配或更新现有的分配时，请指定以下值。

有关使用 CloudFront 控制台创建或更新分配的更多信息，请参阅the section called “创建分配”或the section called “更新分配”。

主题

- 源设置
- 缓存行为设置

- 分配设置
- 自定义错误页面和错误缓存
- 地理限制

源设置

当您使用 CloudFront 控制台创建或更新分配时，您可提供有关一个或多个位置（称为源）的信息，它们是您存储原始版 Web 内容的地方。CloudFront 从您的源获取 Web 内容，并通过遍布全球的边缘服务器网络将其提供给查看器。

有关您可以为分配创建的源的当前最大数量或要请求提高限额，请参阅the section called “分配的一般配额”。

如果您想删除源，您首先必须编辑或删除与该源有关的缓存行为。

> **Important**
>
> 如果您删除源，请确认该源先前提供的文件在另一源中可用，且您的缓存行为将这些文件的请求路由到新的源。

当您创建或更新分配时，您可为每个源指定以下值。

主题

- 源域
- 协议（仅自定义源）
- 源路径
- 名称
- 源访问（仅限 Amazon S3 源）
- 添加自定义标头
- 启用 Origin Shield
- 连接尝试次数
- 连接超时
- 响应超时（仅自定义源）
- 源保持连接超时（仅自定义源）

- 响应和保持连接超时限额

源域

源域是您希望 CloudFront 从中获取此源的对象的 Amazon S3 存储桶或 HTTP 服务器的 DNS 名称，例如：

- Amazon S3 存储桶 – *DOC-EXAMPLE-BUCKET*.s3.*us-west-2*.amazonaws.com

> **Note**
>
> 如果您最近创建了 S3 存储桶，则 CloudFront 分配最长可能需要 24 小时才能返回 HTTP `307 Temporary Redirect` 响应。S3 存储桶名称最长可能需要 24 小时才能传播到所有 AWS区域。在传播完成后，分配将自动停止发送这些重定向响应；您无需执行任何操作。有关更多信息，请参阅为什么我会从 Amazon S3 获得 HTTP 307 临时重定向响应？和临时请求重定向。

- 配置为网站的 Amazon S3 存储桶 – *DOC-EXAMPLE-BUCKET*.s3-website.*us-west-2*.amazonaws.com
- MediaStore 容器 – *examplemediastore*.data.mediastore.*us-west-1*.amazonaws.com
- MediaPackage 端点 – *examplemediapackage*.mediapackage.*us-west-1*.amazonaws.com
- Amazon EC2 实例 – *ec2-203-0-113-25*.compute-1.amazonaws.com
- Elastic Load Balancing 负载均衡器 – *example-load-balancer-1234567890*.*us-west-2*.elb.amazonaws.com
- 您自己的 Web 服务器 – `https://www.example.com`

在源域字段中选择域名或键入域名。域名不区分大小写。

如果您的源是 Amazon S3 存储桶，请注意以下事项：

- 如果将存储桶配置为网站，请输入存储桶的 Amazon S3 静态网站托管端点；不要从源域字段的列表中选择存储桶名称。该静态网站托管端点显示在 Amazon S3 控制台的属性页面中的静态网站托管下。有关更多信息，请参阅 the section called “使用配置为网站端点的 Amazon S3 存储桶”。
- 如果您已为存储桶配置 Amazon S3 Transfer Acceleration，请不要为源域指定 `s3-accelerate` 端点。
- 如果您要从不同的 AWS 账户使用存储桶，而且该存储桶没有配置为网站，请按以下格式输入名称：

bucket-name.s3.*region*.amazonaws.com

如果存储桶位于美国区域，并且您希望 Amazon S3 将请求路由到位于弗吉尼亚州北部的设施，请使用以下格式：

bucket-name.s3.us-east-1.amazonaws.com

- 文件必须公开可读，除非您使用 CloudFront 源访问控制确保 Amazon S3 中内容的安全。有关访问控制的更多信息，请参阅the section called “限制对 Amazon Simple Storage Service 源的访问”。

> **Important**
>
> 如果源是 Amazon S3 存储桶，存储桶名称必须符合 DNS 命名要求。有关更多信息，请转至《Amazon Simple Storage Service 用户指南》中的桶限制。

当您更改源的源域值时，CloudFront 立即开始复制对 CloudFront 边缘站点的更改。在特定边缘站点中的分配配置得到更新之前，CloudFront 会继续将请求转发到先前的源。该边缘站点中的分配配置一经更新，CloudFront 就开始将请求转发到新的源。

更改源并不要求 CloudFront 用新源中的对象重新填充边缘缓存。只要应用程序中的查看器请求未经更改，CloudFront 就会继续提供边缘缓存中已有的对象，直至每个对象的 TTL 过期或直至很少被请求的对象被逐出。

协议（仅自定义源）

> **Note**
>
> 这仅适用于自定义源。

您希望 CloudFront 在从源获取对象时要使用的协议策略。

选择以下任一值：

- 仅 HTTP：CloudFront 仅使用 HTTP 访问源。

> **Important**
>
> 当源是 Amazon S3 静态网站托管端点时，仅 HTTP 是默认设置，因为 Amazon S3 不支持对静态网站托管端点使用 HTTPS 连接。CloudFront 控制台不支持更改 Amazon S3 静态网站托管终端的此设置。

- 仅 HTTPS：CloudFront 仅使用 HTTPS 访问源。
- 匹配查看器：CloudFront 使用 HTTP 或 HTTPS 与源进行通信，具体取决于查看器请求的协议。请注意，CloudFront 仅缓存对象一次，即使查看器使用 HTTP 和 HTTPS 协议发出请求也是如此。

> **Important**
>
> 对于 CloudFront 转发到该源的 HTTPS 查看器请求，源服务器上的 SSL/TLS 证书中的域名之一必须与为源域指定的域名匹配。否则，CloudFront 将使用 HTTP 状态代码 502（无效网关）响应查看器请求，而不是返回请求的对象。有关更多信息，请参阅 the section called “在 CloudFront 中使用 SSL/TLS 证书的要求”。

主题

- HTTP 端口
- HTTPS 端口
- 最低限度源 SSL 协议

HTTP 端口

> **Note**
>
> 这仅适用于自定义源。

（可选）您可以指定自定义源进行侦听的 HTTP 端口。有效值包括端口 80、443 和 1024 至 65535。默认值为端口 80。

> **Important**
>
> 当源是 Amazon S3 静态网站托管端点时，端口 80 是默认设置，因为 Amazon S3 仅对静态网站托管端点支持端口 80。CloudFront 控制台不支持更改 Amazon S3 静态网站托管终端的此设置。

HTTPS 端口

> **Note**
>
> 这仅适用于自定义源。

（可选）您可以指定自定义源进行侦听的 HTTPS 端口。有效值包括端口 80、443 和 1024 至 65535。默认值为端口 443。当协议设置为仅 HTTP 时，不能为 HTTPS 端口指定值。

最低限度源 SSL 协议

> **Note**
>
> 这仅适用于自定义源。

选择 CloudFront 在建立与您的源的 HTTPS 连接时可以使用的最低限度 TLS/SSL 协议。降低 TLS 协议级别会降低安全性，因此建议您选择您的源支持的最新 TLS 协议。当协议设置为仅 HTTP 时，不能为最低限度源 SSL 协议指定值。

如果您使用 CloudFront API 设置 CloudFront 要使用的 TLS/SSL 协议，则无法设置最低限度协议。而是要指定 CloudFront 可以用于您的源的所有 TLS/SSL 协议。有关更多信息，请参阅《Amazon CloudFront API 参考》中的 OriginSslProtocols。

源路径

如果您希望 CloudFront 从源的目录中请求内容，请以斜杠 (/) 开头输入目录路径。CloudFront 会将目录路径附加到源域的值之后，例如 **`cf-origin.example.com/production/images`**。请不要在路径的末尾添加斜杠 (/)。

例如，假设您已为分配指定以下值：

- 源域 – 名为 **DOC-EXAMPLE-BUCKET** 的 Amazon S3 存储桶
- 源路径 – **/production**
- 备用域名 (CNAME)– **example.com**

当用户在浏览器中输入 example.com/index.html 时，CloudFront 会将请求发送到 DOC-EXAMPLE-BUCKET/production/index.html 的 Amazon S3。

当用户在浏览器中输入 example.com/acme/index.html 时，CloudFront 会将请求发送到 DOC-EXAMPLE-BUCKET/production/acme/index.html 的 Amazon S3。

名称

名称是唯一标识此分配中的这个源的字符串。如果您创建默认缓存行为之外的缓存行为，可以使用在此处指定的名称，以标识您希望 CloudFront 在请求与该缓存行为的路径模式匹配时将请求路由到的源。

源访问（仅限 Amazon S3 源）

> **Note**
>
> 这仅适用于 Amazon S3 存储桶源（不使用 S3 静态网站端点的那些源）。

如果您想将对 Amazon S3 存储桶源的访问限制为仅限特定的 CloudFront 分配，请选择来源访问控制设置(推荐)。

如果 Amazon S3 存储桶源可公开访问，请选择公有。

有关更多信息，请参阅 the section called “限制对 Amazon Simple Storage Service 源的访问”。

有关如何要求用户仅使用 CloudFront URL 访问自定义源上的对象的信息，请参阅the section called “在自定义源上限制对文件的访问”。

添加自定义标头

如果您希望 CloudFront 在将请求发送到源时添加自定义标头，请指定标头名称及其值。有关更多信息，请参阅 the section called “向源请求添加自定义标头”。

有关可添加的自定义标头的当前最大数量、自定义标头名称和值的最大长度，以及所有标头名称和值的最大总长度，请参阅配额。

启用 Origin Shield

选择是以启用 CloudFront Origin Shield。有关 Origin Shield 的更多信息，请参阅the section called “使用 Origin Shield”。

连接尝试次数

您可以设置 CloudFront 尝试连接到源的次数。您可以指定 1、2 或 3 作为尝试次数。默认次数（如果您没有另行指定）为 3。

将此设置与连接超时一起使用，可指定 CloudFront 在尝试连接到辅助源或向查看器返回错误响应之前等待多长时间。默认情况下，CloudFront 在尝试连接到辅助源或返回错误响应之前等待长达 30 秒（3 次尝试，每次 10 秒）。您可以通过指定更少的尝试次数、更短的连接超时或两者来减少此时间。

如果指定次数的连接尝试失败，CloudFront 执行以下操作之一：

- 如果源是源组的一部分，CloudFront 尝试连接到辅助源。如果指定次数的连接尝试失败，则 CloudFront 向查看器返回错误响应。
- 如果源不是源组的一部分，则 CloudFront 向查看器返回错误响应。

对于自定义源（包括配置了静态网站托管的 Amazon S3 存储桶），此设置还指定 CloudFront 尝试从源获取响应的次数。有关更多信息，请参阅 the section called “响应超时（仅自定义源）”。

连接超时

连接超时是 CloudFront 在尝试建立与源的连接时等待的秒数。您可以指定 1 到 10 之间（含）的秒数。默认超时（如果您没有另行指定）为 10 秒。

将此设置与连接尝试次数一起使用，可指定 CloudFront 在尝试连接到辅助源或向查看器返回错误响应之前等待多长时间。默认情况下，CloudFront 在尝试连接到辅助源或返回错误响应之前等待长达 30 秒（3 次尝试，每次 10 秒）。您可以通过指定更少的尝试次数、更短的连接超时或两者来减少此时间。

如果 CloudFront 未在指定的秒数内建立到源的连接，CloudFront 执行以下操作之一：

- 如果指定的连接尝试次数超过 1，CloudFront 将再次尝试建立连接。CloudFront 最多尝试 3 次，具体取决连接尝试次数的值。
- 如果所有连接尝试都失败，并且源是源组的一部分，则 CloudFront 尝试连接到辅助源。如果指定次数的连接尝试失败，则 CloudFront 向查看器返回错误响应。
- 如果所有连接尝试都失败，且源不是源组的一部分，则 CloudFront 向查看器返回错误响应。

响应超时（仅自定义源）

来源响应超时（也称为来源读取超时 或来源请求超时）适用于以下两个值：

- CloudFront 在将请求转发到源后等待响应的时间长度（以秒为单位）。
- CloudFront 从收到来自源的一个响应数据包到收到下一个数据包之间等待的时间长度（以秒为单位）。

> **Tip**
>
> 如果您由于查看器遇到了 HTTP 504 状态代码错误而希望增加超时值，请考虑探索其他方法来在更改超时值之前消除这些错误。请参阅 the section called “HTTP 504 状态代码（网关超时）” 中的故障排除建议。

CloudFront 行为取决于查看器请求中的 HTTP 方法：

- `GET` 和 `HEAD` 请求 – 如果源在响应超时的持续时间内没有响应或停止响应，则 CloudFront 中断连接。CloudFront 根据 the section called “连接尝试次数” 的值再次尝试进行连接。
- `DELETE`、`OPTIONS`、`PATCH`、`PUT` 和 `POST` 请求 – 如果源在读取超时期间未响应，CloudFront 将中断连接并且不会再次尝试联系源。如有必要，客户端可以重新提交请求。

源保持连接超时（仅自定义源）

保持连接超时是 CloudFront 在获取最后一个响应数据包后，尝试保持与自定义源的连接的时间长度（以秒为单位）。保持持久性连接可节省重新建立 TCP 连接和执行后续请求的另一个 TLS 握手所需的时间。增加保持连接超时有助于改善分配的每连接请求指标。

> **Note**
>
> 为了让源保持连接超时值生效，源必须配置为允许持久性连接。

响应和保持连接超时限额

> **Note**
>
> 这仅适用于自定义源。

- 响应超时的默认值为 30 秒。
- 保持连接超时的默认值为 5 秒。
- 对于任一限额，您可以指定 1 到 60 秒内的值。要申请提升限额，请在 AWS Support Center Console 中创建案例。

在为您的 AWS 账户请求延长超时后，请更新分配源，使其具有您所需的响应超时和保持连接超时值。为账户提升限额并不会自动更新您的源。例如，如果您使用 Lambda@Edge 函数将保持连接超时设置为 90 秒，则您的源的保持连接超时值必须为 90 秒或更长时间。否则，您的 Lambda@Edge 函数可能会无法执行。

有关分配限额的更多信息，请参阅分配的一般配额。

缓存行为设置

通过设置缓存行为，您可以为您网站上文件的特定 URL 路径模式配置各种 CloudFront 功能。例如，一个缓存行为可能适用于您用作 CloudFront 源服务器的 Web 服务器上 `images` 目录中所有的 `.jpg` 文件。您可为每个缓存行为配置的功能包括：

- 路径模式
- 如果您已经为 CloudFront 分配配置了多个源，则为您希望 CloudFront 将您的请求转发到的源
- 是否将查询字符串转发到源
- 访问指定文件是否需要签名的 URL
- 是否要求用户使用 HTTPS 访问这些文件
- 这些文件保留在 CloudFront 缓存中的最短时间，不管您的源添加到文件中的任何 `Cache-Control` 标头的值

当您创建新分配时，您为默认缓存行为指定设置，这将自动把所有请求转发到您创建分配时指定的源中。在创建分配后，您可创建其他缓存行为，该行为定义 CloudFront 在其接受与路径模式相匹配的

对象请求时如何响应，例如 `*.jpg`。如果您创建其他缓存行为，默认缓存行为始终是最后一个被处理的。其他缓存行为以他们在 CloudFront 控制台中所列的顺序进行处理，或者，如果您使用 CloudFront API，以他们在分配 `DistributionConfig` 元素中所列的顺序进行处理。有关更多信息，请参阅 路径模式。

当您创建缓存行为是，可指定您希望 CloudFront 从中获取对象的一个源。因此，如果您希望 CloudFront 分配所有源的对象，必须至少具有与您所拥有的源一样多的缓存行为（包括默认缓存行为）。例如，如果您有两个源但只有一个默认缓存行为，则默认缓存行为将导致 CloudFront 从其中一个源获取对象，但永远不会使用另一个源。

有关您可以添加到分配的当前最大缓存行为数，或要请求提高配额（以前称为限制），请参阅分配的一般配额。

主题

- 路径模式
- 源或源组
- 查看器协议策略
- 允许的 HTTP 方法
- 字段级加密 Config
- 缓存的 HTTP 方法
- 基于选择的请求标头进行缓存
- 允许列表标头
- 对象缓存
- 最小 TTL
- 最大 TTL
- 默认 TTL
- 转发 Cookie
- 允许列表 Cookie
- 查询字符串转发和缓存
- 查询字符串允许列表
- Smooth Streaming
- 限制查看器访问（使用签名 URL 或签名 Cookie）

- 可信签署人
- AWS 账户 数字
- 自动压缩对象
- CloudFront 事件
- Lambda 函数 ARN
- 包含正文

路径模式

路径模式（例如，`images/*.jpg`）指定您希望该缓存行为所适用的请求。CloudFront 收到查看器请求时，会按照缓存行为在分配中列出的顺序，将请求路径与路径模式进行比较。第一个匹配者决定了哪个缓存行为适用于该请求。例如，假设您具有以下三个路径模式的三个缓存行为，顺序如下：

- `images/*.jpg`
- `images/*`
- `*.gif`

> **Note**
>
> 可以选择在路径模式的开头包含斜杠 (/)，例如 `/images/*.jpg`。无论开头是否有 /，CloudFront 行为都相同。如果您未在路径的开头指定 /，则会自动隐含此字符；无论开头是否有 /，CloudFront 都会以相同方式处理该路径。例如，CloudFront 将 `/*product.jpg` 视为与 `*product.jpg` 相同

文件 `images/sample.gif` 的请求不符合第一个路径模式，因此，相关的缓存行为不适用于该请求。文件符合第二路径模式，因此，与第二个路径模式有关的缓存行为适用，即使该请求与第三个路径模式也匹配。

> **Note**
>
> 在创建新的分配时，默认缓存行为的路径模式的值将设置为 *（所有文件）且无法更改。此值导致 CloudFront 将您对象的所有请求转发到您在 源域 字段中指定的源。如果对象的请求与任何其他缓存行为的路径模式都不匹配，CloudFront 将应用您在默认缓存行为中指定的行为。

> **Important**
>
> 请谨慎定义路径模式及其顺序，否则您可能会向用户提供非预期的内容访问权限。例如，假设请求匹配两个缓存行为的路径模式。第一个缓存行为不要求签名 URL，而第二个缓存行为要求签名 URL。用户无需使用签名 URL 即可访问对象，因为 CloudFront 处理与第一个匹配有关的缓存行为。

如果您正在使用 MediaPackage 通道，则必须为您为源的端点类型定义的缓存行为包含特定的路径模式。例如，对于 DASH 端点，为路径模式键入 `*.mpd`。有关更多信息和具体说明，请参阅提供使用 AWS Elemental MediaPackage 格式化的实时视频。

您指定的路径适用于指定目录以及该指定目录下所有子目录中所有文件的请求。在评估路径模式时，CloudFront 不会考虑查询字符串或 Cookie。例如，如果 `images` 目录包含 `product1` 和 `product2` 子目录，此路径模式 `images/*.jpg` 适用于 `images`、`images/product1` 和 `images/product2` 目录中任何 .jpg 文件的请求。如果您希望将不同的缓存行为应用于 `images/product1` 目录中的文件而非 `images` 和 `images/product2` 目录中文件，为 `images/product1` 创建单独的缓存行为，并将该缓存行为移到 `images` 目录中缓存行为上方 (前面) 的位置。

您可在路径模式中使用以下通配符：

- `*` 匹配 0 或多个字符。
- `?` 精准匹配 1 个字符。

以下示例展示了通配符如何工作：

路径模式	与路径模式匹配的文件
`*.jpg`	所有 .jpg 文件。
`images/*.jpg`	`images` 目录及该 `images` 目录下子目录中的所有 .jpg 文件。
`a*.jpg`	• 文件名以 a 开头的所有 .jpg 文件，例如，`apple.jpg` 和 `appalachian_trail_2012_05_21.jpg` 。

路径模式	与路径模式匹配的文件
	• 文件路径以 a 开头的所有 .jpg 文件，例如，abra/cadabra/magic.jpg 。
a??.jpg	文件名以 a 开头且随后紧跟两个其他字符的所有 .jpg 文件，例如，ant.jpg 和 abe.jpg。
.doc	文件扩展名以 .doc 开头的所有文件，例如，.doc、.docx 和 .docm 文件。在这种情况下，您不能使用路径模式 *.doc?，因为该路径模式不适用于 .doc 文件的请求；? 通配符精确替换一个字符。

路径模式的长度上限是 255 个字符。该值可包含以下任何字符：

- A-Z, a-z

  路径模式区分大小写，因此，路径模式 *.jpg 不适用于文件 LOGO.JPG。
- 0-9
- _ - . * $ / ~ " ' @ : +
- &，作为 & 传递和返回

路径规范化

CloudFront 对 URI 路径进行了与 RFC 3986 一致的规范化，然后将该路径与正确的缓存行为进行匹配。缓存行为匹配后，CloudFront 会将原始 URI 路径发送到源。如果它们不匹配，则请求会改为与您的默认缓存行为进行匹配。

会对某些字符进行规范化并从路径中删除它们，例如多个斜杠（//）或句点（..）。这可能会更改 CloudFront 用于匹配预期缓存行为的 URL。

Example 示例

您可以为缓存行为指定 /a/b* 和 /a* 路径。

- 发送 /a/b?c=1 路径的查看器将与 /a/b* 缓存行为进行匹配。

- 发送 /a/b/..?c=1 路径的查看器将与 /a* 缓存行为进行匹配。

要解决规范化路径的问题，您可以更新请求路径或缓存行为的路径模式。

源或源组

此设置仅在您为现有分配创建或更新缓存行为时才适用。

输入现有源或源组的值。该值标识当请求（如 https://example.com/logo.jpg）与缓存行为（如 *.jpg）或默认缓存行为 (*) 的路径模式匹配时，您希望 CloudFront 将请求路由至的源或源组。

查看器协议策略

选择您希望查看器用来访问 CloudFront 边缘站点中的内容的协议策略：

- HTTP 和 HTTPS：查看器可使用两种协议。
- 将 HTTP 重定向到 HTTPS：查看器可使用两种协议，但 HTTP 请求将自动重定向到 HTTP 请求。
- 仅 HTTPS：如果查看器使用了 HTTPS，则只能访问您的内容。

有关更多信息，请参阅 要求在查看器和 CloudFront 之间使用 HTTPS 进行通信。

允许的 HTTP 方法

指定您希望 CloudFront 处理并转发到源的 HTTP 方法：

- GET、HEAD：您只能使用 CloudFront 获取您源中的对象或获取对象标头。
- GET、HEAD、OPTIONS：您只能使用 CloudFront 从源获取对象、获取对象标头或检索您的源服务器支持的选项列表。
- GET、HEAD、OPTIONS、PUT、POST、PATCH、DELETE：您可以使用 CloudFront 获取、添加、更新和删除对象以及获取对象标头。此外，您可以执行其他 POST 操作，例如从 Web 表格提交数据。

> **Note**
>
> CloudFront 将缓存对 GET 和 HEAD 请求以及（可选）OPTIONS 请求的响应。对 OPTIONS 请求的响应与对 GET 和 HEAD 请求的响应单独缓存（OPTIONS 方法包含在 OPTIONS 请求的缓存键中）。CloudFront 不缓存对使用其他方法的请求的响应。

Important

如果您选择 GET, HEAD, OPTIONS 或 GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE，则可能需要限制对 Amazon S3 存储桶或自定义源的访问，以阻止用户执行您不希望他们执行的操作。以下示例说明了如何限制访问：

- 如果将 Amazon S3 作为您的分配的源：创建 CloudFront 源访问控制，以限制对您的 Amazon S3 内容的访问，并为源访问控制授予权限。例如，如果您只是 因为想要使用 PUT 而将 CloudFront 配置为接受并转发这些方法，则仍必须配置 Amazon S3 存储桶策略以适当处理 DELETE 请求。有关更多信息，请参阅 限制对 Amazon Simple Storage Service 源的访问。
- 如果您使用自定义源：配置源服务器以处理所有方法。例如，您将 CloudFront 配置为接受并转发这些方法只是因为您想要使用 POST，则仍必须配置您的源服务器以适当处理 DELETE 请求。

字段级加密 Config

如果您要对特定数据字段强制实施字段级加密，请在下拉列表中选择一个字段级加密配置。

有关更多信息，请参阅 使用字段级加密帮助保护敏感数据。

缓存的 HTTP 方法

指定您是否希望 CloudFront 在查看器提交 OPTIONS 请求时缓存来自源的响应。CloudFront 始终缓存对 GET 和 HEAD 请求的响应。

基于选择的请求标头进行缓存

指定您是否要 CloudFront 基于指定标头的值缓存对象：

- 无(改进缓存) – CloudFront 不根据标头值缓存您的对象。
- 允许列表 – CloudFront 仅根据指定标头的值缓存您的对象。使用允许列表标头选择您希望 CloudFront 进行缓存时所基于的标头。
- 所有 – CloudFront 不缓存与该缓存行为关联的对象。相反，CloudFront 会将每个请求发送到源。（不建议用于 Amazon S3 源。）

无论您选择哪个选项，CloudFront 都会将特定标头转发到您的源并根据您转发的标头执行特定操作。有关 CloudFront 如何处理标头转发的更多信息，请参阅HTTP 请求标头和 CloudFront 行为（自定义源和 Amazon S3 源）。

有关如何使用请求标头在 CloudFront 中配置缓存的更多信息，请参阅根据请求标头缓存内容。

允许列表标头

此设置仅在您为基于选择的请求标头进行缓存选择允许列表时才适用。

指定您希望 CloudFront 在缓存对象时考虑的标头。从可用标头列表中选择标头，然后选择添加。要转发自定义标头，请在字段中输入标头的名称，然后选择添加自定义标头。

有关您可以为每个缓存行为列入允许列表的当前最大标头数，或者要请求提高配额（以前称为限制），请参阅标头的配额。

对象缓存

如果源服务器将 `Cache-Control` 标头添加到对象以控制在 CloudFront 缓存中保留对象的时间长度，并且您不希望更改 `Cache-Control` 值，请选择使用源缓存标头。

要指定在 CloudFront 缓存中保留对象的最短时间和最长时间（不管 `Cache-Control` 标头如何）以及在缺少 `Cache-Control` 标头时在 CloudFront 缓存中保留对象的默认时间，请选择自定义。然后，在最小 TTL、默认 TTL 和最大 TTL 字段中指定值。

有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

最小 TTL

指定您希望对象在 CloudFront 缓存中保留的最短时间（以秒为单位），在此时间之后，CloudFront 会向源发送另一个请求以确定此对象是否已更新。

有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

最大 TTL

指定您希望在 CloudFront 查询您的源以了解对象是否已更新之前，对象保留在 CloudFront 缓存中的最大时长（以秒为单位）。您为最大 TTL 指定的值仅在源向对象添加 HTTP 标头 (例如 `Cache-Control max-age`、`Cache-Control s-maxage` 或 `Expires`) 时适用。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

要指定 Maximum TTL 的值，您必须为 Object Caching 设置选择 Customize 选项。

最大 TTL 的默认值为 31536000 秒（一年）。如果您将最小 TTL 或默认 TTL 的值更改为一个大于 31536000 秒的值，则最大 TTL 的默认值将变为默认 TTL 的值。

默认 TTL

指定您希望对象在 CloudFront 缓存中保留的默认时间（以秒为单位），在此时间之后，CloudFront 会向您的源转发另一个请求以确定此对象是否已更新。您为默认 TTL 指定的值仅在源不向对象添加 HTTP 标头（例如 `Cache-Control max-age`、`Cache-Control s-maxage` 或 `Expires`）时才适用。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

要指定默认 TTL 的值，您必须为对象缓存设置选择自定义选项。

默认 TTL 的默认值为 86400 秒（一天）。如果您将最小 TTL 的值更改为一个大于 86400 秒的值，则默认 TTL 的默认值将更改为最小 TTL 的值。

转发 Cookie

> **Note**
>
> 对于 Amazon S3 源，此选项仅适用于配置为网站端点的存储桶。

指定是否希望 CloudFront 转发 Cookie 到您的源服务器，以及指定转发哪些 Cookie。如果您选择仅转发选定的 Cookie（Cookie 允许列表），请在允许列表 Cookie 字段中输入 Cookie 名称。如果选择全部，不管您的应用程序使用多少，CloudFront 都会转发所有 Cookie。

Amazon S3 不处理 Cookie，且将 Cookie 转发到来源会降低缓存能力。对于将请求转发到 Amazon S3 源的缓存行为，请对转发 Cookie 选择无。

有关转发 cookie 到源的更多信息，请转至 根据 Cookie 缓存内容。

允许列表 Cookie

> **Note**
>
> 对于 Amazon S3 源，此选项仅适用于配置为网站端点的存储桶。

如果您在转发 Cookie 列表中选择允许列表，请在允许列表 Cookie 字段中输入您希望 CloudFront 为该缓存行为转发到原始服务器的 Cookie 的名称。在新一行中输入每个 Cookie 的名称。

可以使用以下通配符来指定 Cookie 名称：

- * 匹配 Cookie 名称中的 0 个或多个字符
- ? 与 Cookie 名称中的 1 个字符完全匹配

例如，假设对象的查看器请求包含一个带以下名称的 Cookie：

`userid_`*`member-number`*

其中，您的每个用户均具有一个唯一的 *`member-number`* 值。您希望 CloudFront 为每个成员缓存对象的单独版本。您可以通过将所有 Cookie 转发到源来完成该操作，但查看器请求包含一些您希望 CloudFront 不要缓存的 Cookie。或者，您也可以指定以下值以作为 Cookie 名称，这会导致 CloudFront 将所有以 `userid_` 开头的 Cookie 转发到源：

```
userid_*
```

有关您可以为每个缓存行为列入允许列表的当前最大 cookie 名称数，或者要请求提高配额（以前称为限制），请参阅Cookies 的配额（旧缓存设置）。

查询字符串转发和缓存

CloudFront 可根据查询字符串参数的值来缓存不同版本的内容。请选择以下任一选项：

无(改进缓存)

如果无论查询字符串参数的值如何，源都返回相同版本的对象，请选择该选项。这增加了 CloudFront 可从缓存满足请求的可能性，这提高了性能并降低了源的负载。

全部转发，基于允许列表进行缓存

如果源服务器根据一个或多个查询字符串参数返回不同版本的对象，请选择该选项。然后，在 查询字符串允许列表 字段中指定您希望 CloudFront 作为缓存基础的参数。

全部转发，基于所有进行缓存

如果源服务器为所有查询字符串参数返回不同版本的对象，请选择该选项。

有关基于查询字符串参数进行缓存的更多信息（包括如何改进性能），请参阅根据查询字符串参数缓存内容。

查询字符串允许列表

此设置仅在您为 查询字符串转发和缓存 选择全部转发，基于允许列表进行缓存时才适用。您可以指定您希望 CloudFront 用作缓存基础的查询字符串参数。

Smooth Streaming

如果您希望以 Microsoft Smooth Streaming 格式分发媒体文件并且您没有 IIS 服务器，则选择是。

如果您有 Microsoft IIS 服务器并且要将其用作以 Microsoft Smooth Streaming 格式分发媒体文件的源，或者您不打算分发 Smooth Streaming 媒体文件，则选择否。

> **Note**
>
> 如果您指定是，您仍可以使用该缓存行为分配其他内容，只要该内容与路径模式值匹配。

有关更多信息，请参阅 为 Microsoft Smooth Streaming 配置点播视频。

限制查看器访问（使用签名 URL 或签名 Cookie）

如果您希望与该缓存行为的 `PathPattern` 匹配的对象请求使用公共 URL，请选择 No。

如果您希望与该缓存行为的 `PathPattern` 匹配的对象请求使用签名 URL，请选择 Yes。然后指定您希望用于创建签名 URL 的 AWS 账户；这些账户被称为可信签署人。

有关可信签署人的更多信息，请参阅指定可以创建签名 URL 和签名 Cookie 的签署人。

可信签署人

仅在您为限制查看器访问（使用签名的 URL 或签名的 Cookie）选择是时才适用。

选择您希望用作此缓存行为的可信签署人的 AWS 账户：

- 本身：将您当前登录到 AWS Management Console的账户作为可信签署人。如果您目前作为 IAM 用户登录，添加相关的 AWS 账户作为可信签署人。
- 指定账户：在 AWS 账号字段中输入可信签署人的账户号码。

要创建可信签署人 URL，AWS 账户必须至少具有一个有效的 CloudFront 密钥对。

> **Important**
>
> 如果您更新您已经用于分发内容的分配，则在准备开始产生对象的签名 URL 时只添加可信签署人。在您把可信签署人添加到分配为后，用户必须使用签名 URL 访问与该缓存行为 `PathPattern` 匹配的对象。

AWS 账户 数字

仅在您为可信签署人选择指定账户时才适用。

除当前账户之外或替代当前账户，如果您想使用 AWS 账户 创建签名 URL，请在该字段每一行输入一个 AWS 账户 号。请注意以下几点：

- 您指定的账户必须至少具有一个有效的 CloudFront 密钥对。有关更多信息，请参阅 为签署人创建密钥对。
- 您不能为 IAM 用户创建 CloudFront 密钥对，因此，不能使用 IAM 用户作为可信签署人。
- 有关如何获得账户的 AWS 账户账号的信息，请参阅《Amazon Web Services 一般参考》中的您的 AWS 账户标识符。
- 如果您输入当前账户的账户号码，CloudFront 则自动选中本身复选框，并从 AWS 账号列表中删除账号。

自动压缩对象

如果您希望 CloudFront 在查看器支持压缩内容时自动压缩某些类型的文件，请选择是。当 CloudFront 压缩内容时，由于文件更小，因此下载速度更快，并会更快地为用户呈现网页。有关更多信息，请参阅提供压缩文件。

CloudFront 事件

此设置适用于 Lambda 函数关联。

您可以选择在以下一个或多个 CloudFront 事件发生时，运行 Lambda 函数：

- 在 CloudFront 收到查看器的请求时（查看器请求）
- 在 CloudFront 将请求转发到源之前（源请求）
- 在 CloudFront 收到来自源的响应时（源响应）

- 在 CloudFront 将响应返回到查看器之前（查看器响应）

有关更多信息，请参阅 决定使用哪个 CloudFront 事件来触发 Lambda@Edge 函数。

Lambda 函数 ARN

此设置适用于 Lambda 函数关联。

指定要为其添加触发器的 Lambda 函数的 Amazon 资源名称（ARN）。要了解如何获取函数的 ARN，请参阅使用 CloudFront 控制台添加触发器过程的步骤 1。

包含正文

此设置适用于 Lambda 函数关联。

有关更多信息，请参阅包括正文。

分配设置

以下值适用于整个分配。

主题

- 价格级别
- AWS WAF Web ACL
- 备用域名 (CNAME)
- SSL 证书
- 自定义 SSL 客户端支持
- 安全策略（最低 SSL/TLS 版本）
- 支持的 HTTP 版本
- 默认根对象
- 日志记录
- 日志存储桶
- 日志前缀
- Cookie 日志记录
- 启用 IPv6

- 注释
- 分配状态

价格级别

选择与您想为 CloudFront 服务支付的最高价对应的价格级别。默认情况下，CloudFront 从所有 CloudFront 区域的边缘站点提供您的对象。

有关价格级别以及您选择的价格级别如何影响分配的 CloudFront 性能的更多信息，请参阅 CloudFront 定价。

AWS WAF Web ACL

您可以使用 AWS WAF 来保护您的 CloudFront 分配，这是一个 Web 应用程序防火墙，可让您保护您的 Web 应用程序和 API，以在请求到达服务器之前阻止请求。您可以在创建或编辑 CloudFront 分配时为分配启用 AWS WAF。

或者，您可以稍后在 AWS WAF 控制台（https://console.aws.amazon.com/wafv2/）中为特定于您的应用程序的其他威胁配置额外的安全保护。

有关 AWS WAF 的更多信息，请参阅《AWS WAF 开发人员指南》https://docs.aws.amazon.com/waf/latest/developerguide/。

备用域名 (CNAME)

可选。指定您想用于对象 URL 的一个或多个域名，代替您创建分配时 CloudFront 指派的域名。您必须拥有该域名，或有权使用它，您可以通过添加 SSL/TLS 证书来验证这一点。

例如，如果您希望对象的 URL：

```
/images/image.jpg
```

像这样：

```
https://www.example.com/images/image.jpg
```

而不是这样：

```
https://d111111abcdef8.cloudfront.net/images/image.jpg
```

为 www.example.com 添加 CNAME。

> **Important**
>
> 如果将 www.example.com 的 CNAME 添加到您的分配中，还必须执行以下操作：
>
> - 创建（或更新）一条 CNAME 记录，让您的 DNS 服务将对 www.example.com 的查询路由到 d111111abcdef8.cloudfront.net。
> - 将来自可信证书颁发机构 (CA) 的一个证书添加到 CloudFront，该证书中涵盖您添加到分配中的域名 (CNAME)，以来验证您是否被授权使用该域名。
>
> 您必须具有权限才能创建 CNAME 记录，并指定域的 DNS 服务提供商。通常，这意味着您拥有该域，或者您正在为域所有者开发应用程序。

有关您可以添加到分配的当前最大备用域名数，或者要请求提高配额（以前称为限制），请参阅分配的一般配额。

更多有关备用域名的信息，请参阅 通过添加备用域名（CNAME）使用自定义 URL。有关 CloudFront URL 的更多信息，请参阅在 CloudFront 中自定义文件的 URL 格式。

SSL 证书

如果指定一个备用域名来与您的分配结合使用，请选择自定义 SSL 证书，然后，要想验证您是否获得授权使用该备用域名，请选择一个涵盖它的证书。如果您希望查看器使用 HTTPS 访问您的对象，请选择支持此操作的设置。

> **Note**
>
> 在您可以指定自定义 SSL 证书之前，您必须指定一个有效的备用域名。有关更多信息，请参阅使用备用域名的要求和 使用备用域名和 HTTPS：

- 默认 CloudFront 证书 (*.cloudfront.net) – 如果要在您的对象的 URL 中使用 CloudFront 域名，如 https://d111111abcdef8.cloudfront.net/image1.jpg，请选择该选项。
- 自定义 SSL 证书 – 如果要将对象的 URL 中的您自己的域名作为备用域名，例如 https://example.com/image1.jpg，则选择该选项。然后，选择一个涵盖该备用域名的证书。证书列表中可以包括以下任一证书：

- AWS Certificate Manager 提供的证书
- 您从第三方证书颁发机构购买并上传到 ACM 的证书
- 您从第三方证书颁发机构购买并上传到 IAM 证书存储的证书

如果选择此设置，则建议您只在对象 URL 中使用一个备用域名 (https://example.com/logo.jpg)。如果您使用 CloudFront 分配域名 (https://d111111abcdef8.cloudfront.net/logo.jpg)，而客户端使用较旧的不支持 SNI 得查看器，则查看器得响应取决于您为支持的客户端选择的值：

- 所有客户端：当 CloudFront 域名与 SSL/TLS 证书中的域名不匹配时，查看器将显示警告。
- 仅支持服务器名称指示 (SNI) 的客户端：CloudFront 将删除与查看器的连接而不返回对象。

自定义 SSL 客户端支持

仅在您为 SSL 证书选择自定义 SSL 证书（example.com）时才适用。如果为分配指定了一个或多个备用域名和自定义 SSL 证书，请选择您希望 CloudFront 如何处理 HTTPS 请求：

- 支持服务器名称指示 (SNI) 的客户端 - (推荐) – 使用此设置，几乎所有新式 Web 浏览器和客户端都可以连接到分配，因为它们支持 SNI。但是，某些查看器可能会使用较旧的 Web 浏览器或不支持 SNI 的客户端，这意味着他们无法连接到分配。

  要使用 CloudFront API 应用此设置，请在 `SSLSupportMethod` 字段中指定 `sni-only`。在 AWS CloudFormation 中，此字段命名为 `SslSupportMethod`（注意不同的大写）。
- 旧版客户端支持 – 使用此设置，旧版 Web 浏览器和不支持 SNI 的客户端可以连接到分配。但是，此设置会产生额外的月度费用。有关确切的价格，请转到 Amazon CloudFront 定价页面，然后在此页中搜索专用 IP 自定义 SSL。

  要使用 CloudFront API 应用此设置，请在 `SSLSupportMethod` 字段中指定 `vip`。在 AWS CloudFormation 中，此字段命名为 `SslSupportMethod`（注意不同的大写）。

有关更多信息，请参阅 选择 CloudFront 如何处理 HTTPS 请求。

安全策略（最低 SSL/TLS 版本）

指定希望 CloudFront 用于与查看器（客户端）建立 HTTPS 连接的安全策略。安全策略确定两个设置：

- CloudFront 与查看器通信时使用的最低 SSL/TLS 协议。
- CloudFront 可用来加密其返回给查看器的内容的密码

有关安全策略（包括每个策略包含的协议和密码）的更多信息，请参阅查看器和 CloudFront 之间支持的协议和密码。

可用的安全策略取决于您为SSL 证书和自定义 SSL 客户端支持指定的值（称为 CloudFront API 中的 `CloudFrontDefaultCertificate` 和 `SSLSupportMethod`）：

- 在 SSL 证书是默认 CloudFront 证书 (*.cloudfront.net) 时（当 API 中 `CloudFrontDefaultCertificate` 是 `true` 时），CloudFront 将安全策略自动设置为 TLSv1。
- 当 SSL 证书是自定义 SSL 证书 (example.com) 并且 自定义 SSL 客户端支持是支持服务器名称指示 (SNI) 的客户端 -（推荐）时（在 API 中 `CloudFrontDefaultCertificate` 是 `false` 并且 `SSLSupportMethod` 是 `sni-only` 时），您可以从以下安全策略中进行选择：
  - TLSv1.2_2021
  - TLSv1.2_2019
  - TLSv1.2_2018
  - TLSv1.1_2016
  - TLSv1_2016
  - TLSv1
- 当 SSL 证书是 自定义 SSL 证书 (example.com) 并且 自定义 SSL 客户端支持是旧版客户端支持时（在 API 中 `CloudFrontDefaultCertificate` 是 `false` 并且 `SSLSupportMethod` 是 `vip` 时），您可以从以下安全策略中进行选择：
  - TLSv1
  - SSLv3

  在此配置中，TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016 和 TLSv1_2016 安全策略在 CloudFront 控制台或 API 中不可用。如果要使用这些安全策略之一，您可以选择以下选项：
  - 评估您的分配是否需要具有专用 IP 地址的旧版客户端支持。如果查看器支持服务器名称指示 (SNI)，建议您将分配的自定义 SSL 客户端支持设置更新为支持服务器名称指示 (SNI) 的客户端（在 API 中将 `SSLSupportMethod` 设置为 `sni-only`）。这使您能够使用任何可用的 TLS 安全策略，并且还可以降低您的 CloudFront 费用。
  - 如果您必须保留具有专用 IP 地址的旧版客户端支持，则可以通过在 AWS 支持中心创建案例来请求其他 TLS 安全策略（TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016 或 TLSv1_2016）之一。

> **Note**
>
> 在联系 AWS Support 以请求此更改之前，请考虑以下事项：
>
> - 当您将其中一个安全策略（TLSv1.2_2021、TLSv1.2_2019、TLSv1.2_2018、TLSv1.1_2016 或 TLSv1_2016）添加到某个旧版客户端支持分配时，安全策略将应用于您的AWS账户中所有旧版客户端支持分配的所有非 SNI 查看器请求。但是，当查看器将 SNI 请求发送到具有旧版客户端支持的分配时，该分配的安全策略适用。要确保将您所需的安全策略应用于发送到 AWS 账户中所有旧版客户端支持分配的所有查看器请求，请将所需安全策略分别添加到每个分配。
> - 根据定义，新安全策略不支持与旧安全策略相同的密码和协议。例如，如果您选择将分配的安全策略从 TLSv1 升级到 TLSv1.1_2016，则该分配将不再支持 DES-CBC3-SHA 密码。有关每个安全策略支持的密码和协议的更多信息，请参阅查看器和 CloudFront 之间支持的协议和密码。

支持的 HTTP 版本

选择您希望分配在查看器与 CloudFront 通信时支持的 HTTP 版本。

要使查看器和 CloudFront 使用 HTTP/2，查看器必须支持 TLSv1.2 或更高版本以及服务器名称标识 (SNI)。CloudFront 不通过 HTTP/2 为 gRPC 提供原生支持。

要使查看器和 CloudFront 使用 HTTP/3，查看器必须支持 TLSv1.3 或更高版本以及服务器名称标识 (SNI)。CloudFront 支持 HTTP/3 连接迁移，允许查看器在不丢失连接的情况下切换网络。有关连接迁移的更多信息，请参阅 RFC 9000 中的连接迁移。

> **Note**
>
> 有关受支持的 TLSv1.3 密码的更多信息，请参阅查看器和 CloudFront 之间支持的协议和密码。

默认根对象

可选。当查看器请求分发的根 URL (`index.html`) 而不是分发中的对象 (`https://www.example.com/`) 时，您希望 CloudFront 从您的源（例如，`https://www.example.com/product-description.html`）中请求的对象。指定一个默认根对象，以避免公开分配的内容。

名称的长度上限是 255 个字符。该名称可包含以下任何字符：

- A-Z, a-z
- 0-9
- _ - . * $ / ~ " '
- &，作为 & 传递和返回

当您指定默认根对象时，请仅输入对象名称，例如 index.html。不要在对象名称前添加 /。

有关更多信息，请参阅 指定默认根对象。

日志记录

您是否希望 CloudFront 记录对象的每个请求信息并将日志文件存储在 Amazon S3 存储桶中。您可以随时启用或禁用日志记录。启用日志记录无额外的费用，但存储和访问 Amazon S3 存储桶中的文件会产生常规的 Amazon S3 费用。您可以随时删除日志文件。有关 CloudFront 访问日志的更多信息，请参阅配置和使用标准日志（访问日志）。

日志存储桶

如果您为日志记录选择开启，则为您希望 CloudFront 存储访问日志的 Amazon S3 存储桶，例如，*myLogs-*DOC-EXAMPLE-BUCKET.s3.amazonaws.com。

> **Important**
>
> 请勿在以下任何区域中选择 Amazon S3 存储桶，因为 CloudFront 不会将标准日志传送到这些区域中的存储桶：
>
> - Africa (Cape Town)
> - 亚太地区（香港）
> - 亚太地区（海得拉巴）
> - 亚太地区（雅加达）
> - 亚太地区（墨尔本）
> - 加拿大西部（卡尔加里）
> - 欧洲地区（米兰）
> - 欧洲（西班牙）
> - 欧洲（苏黎世）

- 以色列（特拉维夫）
- 中东（巴林）
- 中东（阿联酋）

如果您启用日志记录，CloudFront 记录有关对象每个最终用户请求的信息并将文件存储在指定的 Amazon S3 存储桶中。您可以随时启用或禁用日志记录。有关 CloudFront 访问日志的更多信息，请参阅配置和使用标准日志（访问日志）。

> **Note**
>
> 您必须拥有必需的权限才能获取和更新 Amazon S3 存储桶 ACL，并且存储桶的 S3 ACL 必须向您授予 `FULL_CONTROL`。这允许 CloudFront 提供 `awslogsdelivery` 账户权限，以将日志文件保存在存储桶中。有关更多信息，请参阅 配置标准日志记录和访问您的日志文件所需的权限。

日志前缀

可选。如果您为日志记录选择开启，请指定您希望 CloudFront 为此分配的访问日志文件名添加前缀的字符串（如有），例如 `exampleprefix/`。尾随斜杠 (/) 是可选的，但建议简化浏览您的日志文件。有关 CloudFront 访问日志的更多信息，请参阅配置和使用标准日志（访问日志）。

Cookie 日志记录

如果希望 CloudFront 将 cookie 包含在访问日志中，请选择开启。如果您选择将 cookie 包含在日志中，CloudFront 则记录所有 cookie，而不管您如何配置此分配的缓存行为：转发所有 cookie，不转发 cookie，或将指定的 cookie 列表转发到源。

Amazon S3 不处理 cookie，因此除非您的分配也包括 Amazon EC2 或其他自定义源，否则建议您为 Cookie 日志记录的值选择关闭。

有关 cookie 的更多信息，请转至 根据 Cookie 缓存内容。

启用 IPv6

IPv6 是新版本的 IP 协议。它使用更大的地址空间，最终将取代 IPv4。CloudFront 始终响应 IPv4 请求。如果您希望 CloudFront 响应来自 IPv4 IP 地址（例如 192.0.2.44）的请求和来自 IPv6 地址（例如 2001:0db8:85a3::8a2e:0370:7334）的请求，请选择 Enable IPv6（启用 IPv6）。

通常，如果您有 IPv6 网络上的用户需要访问您的内容，则应启用 IPv6。不过，如果您使用签名 URL 或签名的 Cookie 来限制对您的内容的访问，并且您使用自定义策略（该策略包含 `IpAddress` 参数以限制可访问您的内容的 IP 地址），请不要启用 IPv6。如果您希望按 IP 地址限制对某些内容的访问，而不限制对其他内容的访问 (或限制访问，但不按 IP 地址实施限制)，则您可创建两个分配。有关使用自定义策略创建签名 URL 的信息，请参阅使用自定义策略创建签名 URL。有关使用自定义策略创建签名 Cookie 的信息，请参阅使用自定义策略设置签名 Cookie。

如果您使用 Route 53 别名资源记录集将流量路由到您的 CloudFront 分配，在满足以下两个条件时，您需要创建另一个别名资源记录集：

- 您为分发启用 IPv6
- 您在对象的 URL 中使用备用域名

有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的使用域名将流量路由到 Amazon CloudFront 分配。

如果您已使用 Route 53 或其他 DNS 服务创建 CNAME 资源记录集，则无需进行任何更改。CNAME 记录会将流量路由到您的分配，无论查看器请求的 IP 地址格式如何。

如果您启用 IPv6 和 CloudFront 访问日志，`c-ip` 列将包含 IPv4 和 IPv6 格式的值。有关更多信息，请参阅 配置和使用标准日志（访问日志）。

> **Note**
>
> 为了维护客户的高可用性，CloudFront 将使用 IPv4 响应查看器请求（如果我们的数据表明 IPv4 将提供更佳的用户体验）。要查明 CloudFront 通过 IPv6 提供的请求的百分比，请为分配启用 CloudFront 日志记录并解析 `c-ip` 列，该列包含已发出请求的查看器的 IP 地址。此百分比将随时间增大，但仍只占据流量的一小部分，因为 IPv6 并未在全球范围内得到所有查看器网络的支持。一些查看器网络具有很好的 IPv6 支持，但其他查看器网络根本不支持 IPv6。(查看器网络类似于您的家庭 Internet 或无线运营商。)
>
> 有关我们对 IPv6 的支持的更多信息，请参阅 CloudFront 常见问题。有关启用访问日志的信息，请参阅字段 日志记录、日志存储桶 和 日志前缀。

注释

可选。在创建分配时，您最多可以包含 128 字符的注释。您可以随时更新注释。

分配状态

表明您是否希望分配在一经部署时就被启用或禁用：

- 已启用是指，只要分配一经全面部署，您就可部署使用分配域名的链接，并且用户可检索内容。无论何时启用分配，CloudFront 将接受并处理任何最终用户使用与该分配有关的域名对内容的请求。

  当您创建、修改或删除 CloudFront 分配时，需要一定的时间才能将您所做的更改传播到 CloudFront 数据库。即刻发起的对分配相关信息的请求可能不会显示出该更改。传播通常在几分钟内完成，但高系统负载或网络分区可能会延长该时间。
- 已禁用是指，即使分配可能已经部署且准备好使用，用户也不能使用它。无论何时禁用分配，CloudFront 都不接受任何最终用户使用与该分配有关的域名对内容的请求。除非您将分配从禁用切换到启用（通过更新分配的配置），否则任何人都不能使用它。

您可根据您想要的频率在禁用和启用之间转换分配。遵照更新分配配置的过程。有关更多信息，请参阅更新分配。

自定义错误页面和错误缓存

您可以让 CloudFront 在您的 Amazon S3 或自定义源向 CloudFront 返回 HTTP 4xx 或 5xx 状态代码时向查看器返回一个对象（例如 HTML 文件）。您还可以指定从您的源发出的错误响应或自定义错误页面缓存在 CloudFront 边缘缓存中进行缓存的时长。有关更多信息，请参阅 为特定 HTTP 状态代码创建自定义错误页面。

> **Note**
>
> 以下值不包含在“创建分配”向导中，所以您只能在更新分配时配置自定义错误页面。

主题

- HTTP 错误代码
- 响应页面路径
- HTTP 响应代码
- 错误缓存最小 TTL（秒）

HTTP 错误代码

您希望 CloudFront 返回自定义错误页面时所对应的 HTTP 状态代码。您可以将 CloudFront 配置为返回 CloudFront 缓存的无、部分或全部 HTTP 状态代码的自定义错误页面。

响应页面路径

在源返回您为错误代码指定的 HTTP 状态代码（如 403）时，您希望 CloudFront 返回到查看器的自定义错误页面的路径（如 `/4xx-errors/403-forbidden.html`）。如果您希望将您的对象和自定义错误页面存储在不同的位置，您的分配必须包含满足以下条件时的缓存行为：

- Path Pattern 的值与您的自定义错误消息的路径匹配。例如，假设您在 Amazon S3 存储桶中名为 `/4xx-errors` 的目录下为 4xx 错误保存了自定义错误页面。您的分配必须包含缓存行为，其路径模式将对自定义错误页面的请求路由至该位置，例如 /4xx-errors/*。
- 源值指定包含您的自定义错误页面的源的源 ID 值。

HTTP 响应代码

您希望 CloudFront 将其与自定义错误页面一起返回给查看器的 HTTP 状态代码。

错误缓存最小 TTL（秒）

您希望 CloudFront 缓存从源服务器发出的错误响应的最短时间。

地理限制

如果您需要阻止选定国家/地区的用户访问您的内容，可以使用允许列表或阻止列表配置您的 CloudFront 分配。配置地理限制不收取额外费用。有关更多信息，请参阅 限制您的内容的地理分配。

测试分配

在您已经创建分配后，CloudFront 将知晓您的源服务器的位置，而且您也知晓与该分配相关的域名。要测试您的分配，请执行以下操作：

1. 请等待您的分配部署完成。
   - 在控制台中查看您的分配的详细信息。部署分配后，上次修改时间字段将从正在部署更改为部署日期和时间。
2. 使用以下步骤创建链接，指向您的采用 CloudFront 域名的对象。

3. 测试链接。CloudFront 将对象提供给您的网页或应用程序。

创建指向对象的链接

使用以下步骤，为您的 CloudFront Web 分配中的对象创建测试链接。

为 Web 分配中的对象创建链接

1. 将以下 HTML 代码复制到新文件中，用您的分配域名替换“*domain-name*”，然后用您的对象名称替换“*object-name*”。

```
<html>
<head>My CloudFront Test</head>
<body>
<p>My text content goes here.</p>
<p><img src="https://domain-name/object-name" alt="my test image"
</body>
</html>
```

   例如，如果您的域名为 d111111abcdef8.cloudfront.net，且对象名称为 image.jpg，则链接的 URL 将为：

   https://d111111abcdef8.cloudfront.net/image.jpg.

   如果您的对象是在源服务器上的一个文件夹里，则此文件夹也必须包含在 URL 中。例如，如果 image.jpg 位于源服务器上的图像文件夹中，则 URL 为：

   https://d111111abcdef8.cloudfront.net/images/image.jpg

2. 将 HTML 代码保存在具有 .html 文件扩展名的文件中。
3. 在浏览器中打开您的网页，以确保您能够看到您的对象。

浏览器返回带嵌入图像文件的页面，该文件由边缘站点提供，且 CloudFront 确定该边缘站点适合于服务对象。

更新分配

在 CloudFront 控制台中，您可以看到与您 AWS 账户关联的 CloudFront 分配、查看分配的设置以及更新大部分设置。请注意，分配传播到 AWS 边缘站点后您所做的设置更改才会生效。

更新 CloudFront 分配

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择分配的 ID。该列表包含与您登录到 CloudFront 控制台时使用的 AWS 账户关联的所有分配。
3. 要编辑分配的设置，请选择分配设置选项卡。
4. 要更新常规设置，选择编辑。否则，请选择要更新的设置的选项卡：源或行为。
5. 更新后要保存您的更改，请选择是，编辑。有关字段的信息，请参阅以下主题：
   - 常规设置：分配设置
   - 源设置：源设置
   - 缓存行为设置：缓存行为设置
6. 如果要删除分配中的源，请执行以下操作：
   a. 选择行为，并确保您将与源关联的所有默认缓存行为移动到另一个源。
   b. 选择源，然后选择一个源。
   c. 选择 Delete。

您还可以使用 CloudFront API 更新分配：

- 要更新分配，请参阅《Amazon CloudFront API 参考》中的 UpdateDistribution。

> **Important**
>
> 在更新分配时，请注意需要使用很多在创建分配时不需要的其他字段。要帮助确保在使用 CloudFront API 更新分配时包含所有必需字段，请按照《Amazon CloudFront API 参考》内 UpdateDistribution 中所述的步骤进行操作。

当您将更改保存到您的分配配置中时，CloudFront 开始将更改传送到所有边缘站点。后续的配置更改按各自的顺序传播。在您的配置在边缘站点中经更新前，CloudFront 将继续根据先前的配置从该位置提供内容。在您的配置在边缘站点已更新后，CloudFront 将立即根据新的配置从该位置提供内容。

您的更改不会同时传播到每个边缘站点。CloudFront 在传播您的更改时，我们不能确定一个指定边缘站点是根据先前配置还是新配置提供内容。

要查看您的更改何时被传播，请在控制台中查看您的分配的详细信息。上次修改时间字段将从正在部署更改为部署完成的日期和时间。

标记分配

标签是可用于标识和组织 AWS 资源的词或短语。您可以向每个资源添加多个标签，并且每个标签都包含您定义的一个键和一个值。例如，键可能是“domain”，值可能是“example.com”。您可以根据添加的标签搜索和筛选您的资源。

您可以在 CloudFront 中使用标签，例如以下示例：

- 对 CloudFront 分配强制实施基于标签的权限。有关更多信息，请参阅 ABAC 以及 CloudFront。
- 跟踪不同类别的账单信息。在您将标签应用于 CloudFront 分配或其他AWS资源（例如 Amazon EC2 实例或 Amazon S3 存储桶）并激活标签时，AWS 将以逗号分隔值（CSV 文件）格式生成一份成本分配报告，其中包括按活动标签汇总的使用率和成本。

  您可以设置代表业务类别（例如成本中心、应用程序名称或所有者）的标签，以便整理多种服务的成本。有关对成本分配使用标签的更多信息，请参阅《AWS Billing 用户指南》中的使用成本分配标签。

> **注意**
>
> - 您可以为分配添加标签，但不能为源访问身份或失效添加标签。
> - CloudFront 当前不支持标签编辑器和资源组。
> - 有关可以向分配添加的最大标签数，请参阅常规配额。

目录

标签限制

下面是适用于标签的基本限制：

- 要了解每个分配的最大标签数，请参阅常规配额。
- 最大密钥长度 – 128 个 Unicode 字符
- 最大值长度 – 256 个 Unicode 字符
- 键和值的有效值 – a-z、A-Z、0-9、空格和以下字符：_ . : / = + - 和 @
- 标签键和值区分大小写
- 请不要使用 `aws:` 作为键的前缀。此前缀是专为 AWS 使用而预留。

为分配添加、编辑和删除标签

您可以使用 CloudFront 控制台管理分配标签。

为分配添加、编辑或删除标签

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 为要更新的分配选择 ID。
3. 选择 Tags 选项卡。
4. 选择管理标签。
5. 在 Manage tags（管理标签）页面上，可以执行以下操作：
   - 要添加标签，请输入该标签的键和（可选）值。要添加更多标签，请选择添加新标签。
   - 要编辑标签，请更改标签的键或值，或者同时更改两者。您可以删除标签的值，但键是必需的。
   - 要删除标签，请选择删除。
6. 选择保存更改。

编程标记

您也可以使用 CloudFront API、AWS Command Line Interface（AWS CLI）、AWS SDK 和 AWS Tools for Windows PowerShell 来应用标签。有关更多信息，请参阅以下主题：

- CloudFront API 操作：
  - ListTagsForResource
  - TagResource
  - UntagResource

- AWS CLI – 请参阅 AWS CLI 命令参考中的 cloudfront
- AWS SDK – 请参阅 AWS 文档页面上的适用 SDK 文档
- Tools for Windows PowerShell – 请参阅 AWS Tools for PowerShell Cmdlet 参考中的 Amazon CloudFront。

删除分配

以下过程使用 CloudFront 控制台删除分配。有关使用 CloudFront API 进行删除的信息，请参阅《Amazon CloudFront API 参考》中的删除分配。

如果您需要删除 OAC 附加到 S3 存储桶的分配，请参阅删除将其 OAC 附加到 S3 存储桶的分配以获取重要详细信息。

> **Note**
>
> 请注意，您可以删除某个分配之前，须先将其禁用，这需要权限以更新分发。
> 如果禁用的分配包含一个相关的备用域名，则 CloudFront 会停止接受该域名（例如 www.example.com）的流量，即便另一个分配有一个包含通配符 (*) 且匹配同样域的备用域名（例如 *. example.com）。

删除 CloudFront 分配

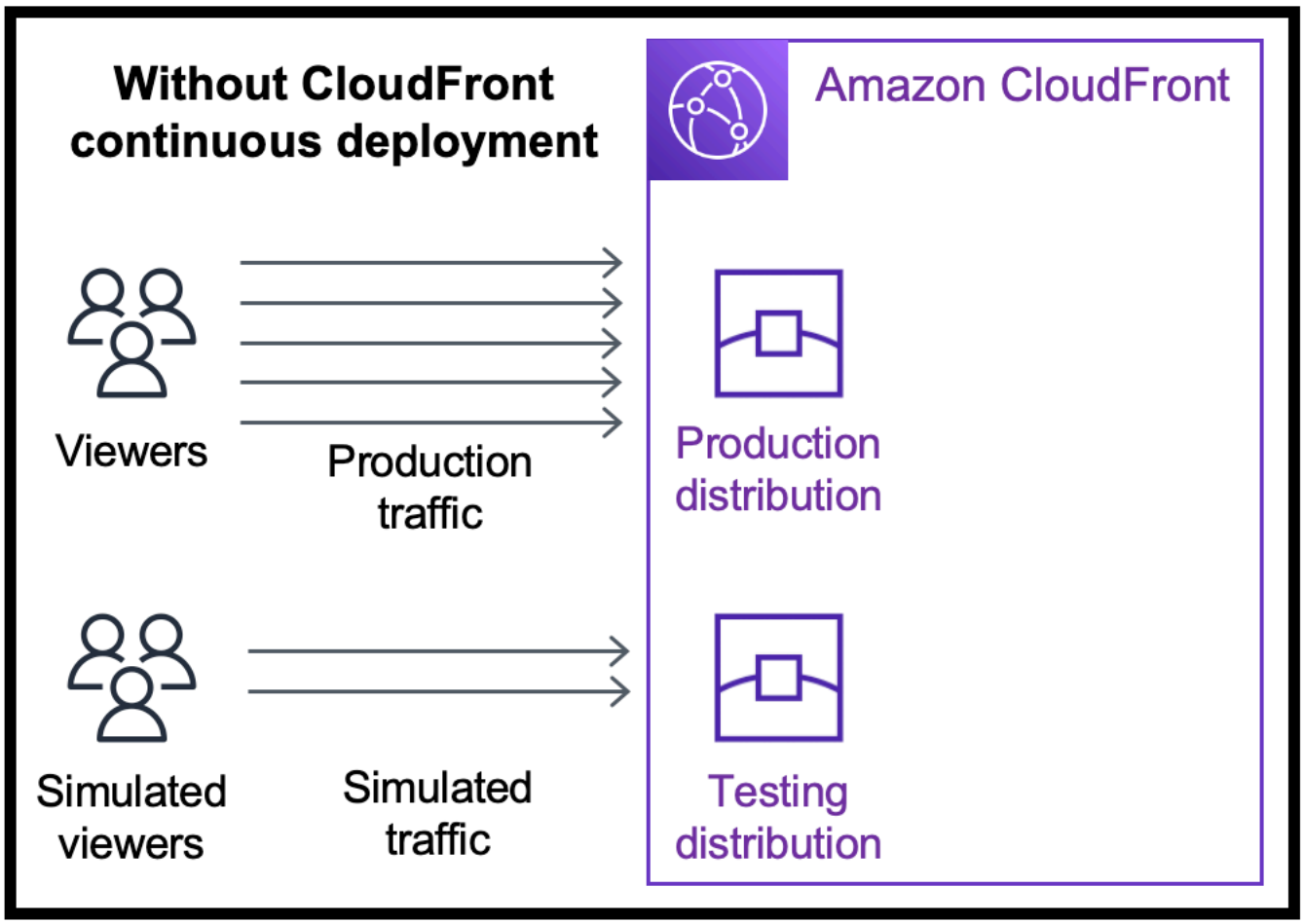
1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在 CloudFront 控制台的右侧窗格中，找到您想要删除的分配。
   - 如果状态列显示已禁用，请跳至步骤 6。
   - 如果状态列显示已启用，但分配的上次修改时间列中仍显示正在部署，请等待部署完成后再继续执行步骤 3。
3. 在 CloudFront 控制台的右侧窗格中，选中要删除的分配的复选框。
4. 选择禁用以禁用分配，并选择是，禁用予以确认。然后选择关闭。
   - 状态列的值立即变为已禁用。
5. 等待，直到新的时间戳显示在上次修改时间列下。
   - CloudFront 将您的更改传播到所有边缘站点可能需要几分钟时间。

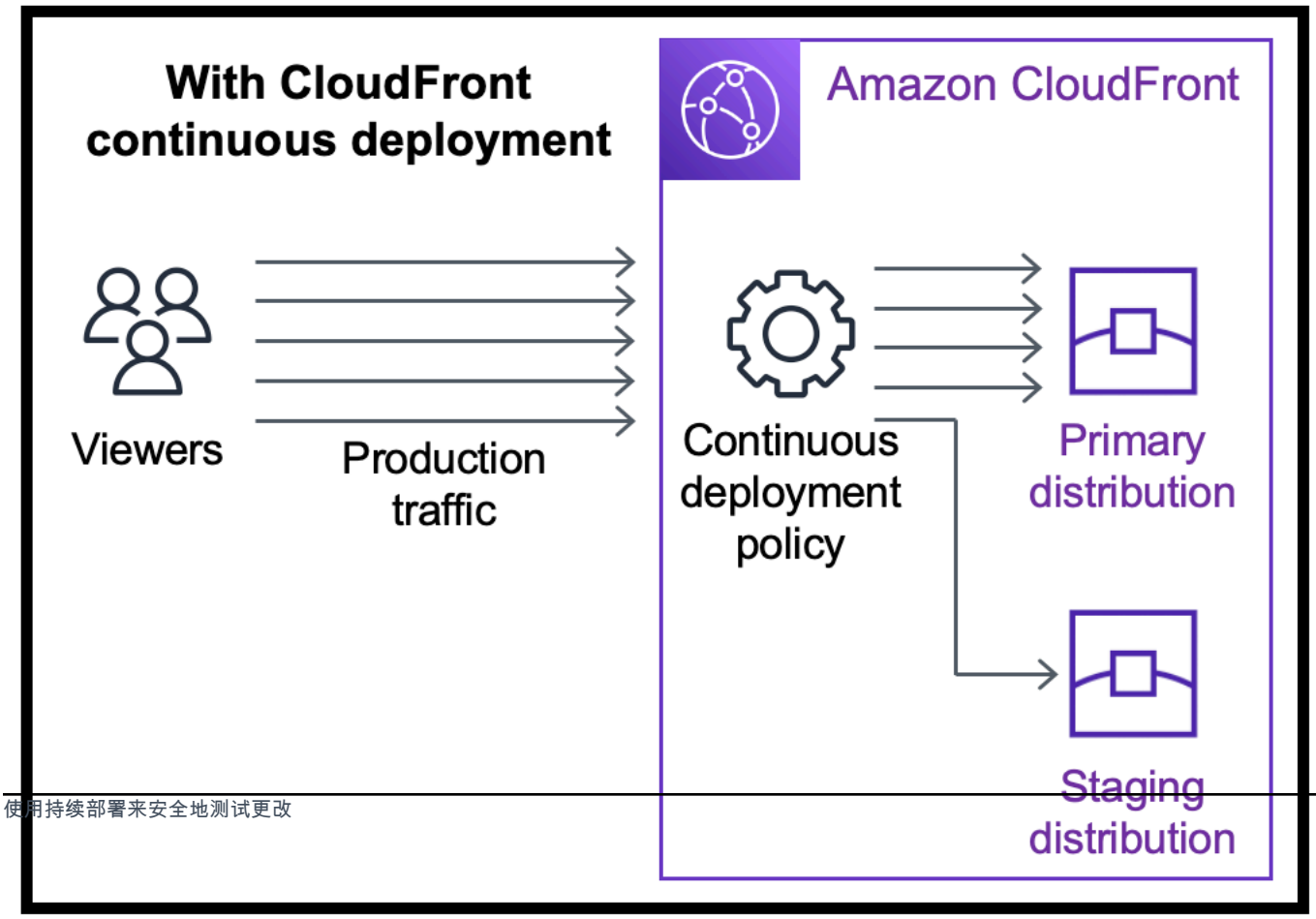
6. 选中要删除的分配对应的复选框。
7. 选择删除，删除。
   - 如果删除选项不可用，表示 CloudFront 仍在将您的更改传播到边缘站点。请等待，直到新的时间戳显示在上次修改时间列下，然后重复步骤 6-7。

使用 CloudFront 持续部署来安全地测试 CDN 配置更改

利用 Amazon CloudFront 持续部署，您可以先对一部分生产流量进行测试，从而安全地部署对 CDN 配置的更改。您可以使用暂存分配和持续部署策略来将一些来自实际（生产）查看器的流量发送到新的 CDN 配置，并验证其是否按预期运行。您可以实时监控新配置的性能，并在准备就绪时提升新配置以通过主分配为所有流量提供服务。

以下图表说明了使用 CloudFront 持续部署的好处。如果没有该部署，您将必须使用模拟流量来测试 CDN 配置更改。利用持续部署，您可以使用一部分生产流量来测试更改，然后在准备就绪时将更改推广到主分配。


Without CloudFront continuous deployment
Amazon CloudFront
Viewers
Production traffic
Production distribution
Simulated viewers
Simulated traffic
Testing distribution


With CloudFront continuous deployment
Amazon CloudFront
Viewers
Production traffic
Continuous deployment policy
Primary distribution
Staging distribution

阅读以下主题，了解有关使用持续部署的更多信息。

主题

CloudFront 持续部署工作流

以下简要工作流程说明了如何通过 CloudFront 持续部署来安全地测试和部署配置更改。

1. 选择要用作主分配的分配。主分配是目前为生产流量提供服务的分发。
2. 在主分配中，创建暂存分配。暂存分配作为主分配的副本开始。
3. 在持续部署策略中创建流量配置，并将其附加到主分配。这将决定 CloudFront 如何将流量路由到暂存分配。有关将请求路由到暂存分配的更多信息，请参阅the section called “将请求路由到暂存分配”。
4. 更新暂存分配的配置。有关您可以更新的设置的更多信息，请参阅the section called “更新主分配和暂存分配”。
5. 监控暂存分配以确定配置更改是否按预期执行。有关监控暂存分配的更多信息，请参阅the section called “监控暂存分配”。

   在监控暂存分配时，您可以：

   - 重新更新暂存分配的配置，以继续测试配置更改。
   - 更新持续部署策略（流量配置），以向暂存分配发送更多或更少流量。
6. 如果您对暂存分配的性能感到满意，请将暂存分配的配置提升为主分配，这会将暂存分配的配置复制到主分配。这还将禁用持续部署策略，这意味着 CloudFront 会将所有流量路由到主分配。

您可以构建自动化来监控暂存分配的性能（步骤 5），并在满足特定条件时自动升级配置（步骤 6）。

提升配置后，可以在下次要测试配置更改时重用相同的暂存分配。

有关在 CloudFront 控制台、AWS CLI 或 CloudFront API 中使用暂存分配和持续部署策略的更多信息，请参阅以下部分。

使用暂存分配和持续部署策略

您可以使用 AWS Command Line Interface（AWS CLI）或 CloudFront API 在 CloudFront 控制台中创建、更新和修改暂存分配与持续部署策略。

使用持续部署策略创建暂存分配

以下步骤演示如何使用持续部署策略创建暂存分配。

Console

您可以使用AWS Management Console，通过持续部署策略创建暂存分配。

创建暂存分配和持续部署策略（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择分配。
3. 选择要用作主分配的分配。主分配是当前为生产流量提供服务的分配，您将从中创建暂存分配。
4. 在持续部署部分中，选择创建暂存分配。这将打开创建暂存分配向导。
5. 在创建暂存分配向导中，执行以下操作：

   a. （可选）键入暂存分配的描述。

   b. 选择下一步。

   c. 修改暂存分配的配置。有关您可以更新的设置的更多信息，请参阅the section called “更新主分配和暂存分配”。

      修改完暂存分配的配置后，选择下一步。

   d. 使用控制台指定流量配置。这将决定 CloudFront 如何将流量路由到暂存分配。（CloudFront 将流量配置存储在持续部署策略中。）

      有关流量配置中的选项的更多信息，请参阅the section called “将请求路由到暂存分配”。

      完成流量配置后，选择下一步。

   e. 查看暂存分配的配置，包括流量配置，然后选择创建暂存分配。

在 CloudFront 控制台中完成创建暂存分配向导后，CloudFront 将执行以下操作：

- 使用您指定的设置创建暂存分配（在步骤 5c 中）
- 使用您指定的流量配置创建持续部署策略（在步骤 5d 中）
- 将持续部署策略附加到您从中创建暂存分配的主分配

当主分配的配置以及附加的持续部署策略部署到边缘站点时，CloudFront 会开始根据流量配置将指定部分的流量发送到暂存分配。

CLI

要使用 AWS CLI 创建暂存分配和持续部署策略，请执行以下步骤。

创建暂存分配（CLI）

1. 将 aws cloudfront get-distribution 和 grep 命令结合使用以获取要用作主分配分发的 ETag 值。主分配是当前为生产流量提供服务的分发，将从中创建暂存分配。

   以下命令是一个示例。在以下示例中，将 *primary_distribution_ID* 替换为主分配的 ID。

   ```
   aws cloudfront get-distribution --id primary_distribution_ID | grep 'ETag'
   ```

   复制 ETag 值，因为您需要它来执行下一个步骤。

2. 使用 aws cloudfront copy-distribution 命令创建暂存分配。以下示例命令使用转义字符（\）和换行符来提高可读性，但您应在该命令中省略这些字符。在以下示例命令中：

   - 将 *primary_distribution_ID* 替换为主分配的 ID。
   - 将 *primary_distribution_ETag* 替换为主分配的 ETag 值（您已在上一步中获得）。
   - （可选）将 *CLI_example* 替换为所需的调用方参考 ID。

   ```
   aws cloudfront copy-distribution --primary-distribution-
   id primary_distribution_ID \
                                    --if-match primary_distribution_ETag \
                                    --staging \
                                    --caller-reference 'CLI_example'
   ```

该命令的输出显示了有关暂存分配及其配置的信息。复制暂存分配的 CloudFront 域名，因为您在下一步中需要使用它。

创建持续部署策略（带输入文件的 CLI）

1. 使用以下命令创建名为 continuous-deployment-policy.yaml 的文件，其中包含 create-continuous-deployment-policy 命令的所有输入参数。以下命令使用转义字符（\）和换行符来提高可读性，但您应在该命令中省略这些字符。

```
aws cloudfront create-continuous-deployment-policy --generate-cli-skeleton yaml-
input \
                                                    > continuous-deployment-
policy.yaml
```

2. 打开刚创建的名为 continuous-deployment-policy.yaml 的文件。编辑该文件以指定所需的持续部署策略设置，然后保存该文件。当您编辑该文件时：

   - 在 StagingDistributionDnsNames 部分中：
     - 将 Quantity 的值更改为 1。
     - 对于 Items，粘贴暂存分配的 CloudFront 域名（您已在上一步中保存）。
   - 在 TrafficConfig 部分中：
     - 选择 Type（SingleWeight 或 SingleHeader）。
     - 删除其他类型的设置。例如，如果您需要基于权重的流量配置，请将 Type 设置为 SingleWeight，然后删除 SingleHeaderConfig 设置。
     - 要使用基于权重的流量配置，请将 Weight 的值设置为介于 .01（百分之一）和 .15（百分之十五）之间的十进制数。

   有关 TrafficConfig 中的选项的更多信息，请参阅the section called “将请求路由到暂存分配”和the section called “基于权重的配置的会话粘性”。

3. 使用以下命令通过 continuous-deployment-policy.yaml 文件中的输入参数创建持续部署策略。

```
aws cloudfront create-continuous-deployment-policy --cli-input-yaml file://
continuous-deployment-policy.yaml
```

复制命令输出中的 `Id` 值。这是持续部署策略 ID，您在下一步中需要它。

将持续部署策略附加到主分配（带输入文件的 CLI）

1. 使用以下命令将主分配的配置保存到一个名为 `primary-distribution.yaml` 的文件中。将 *`primary_distribution_ID`* 替换为主分配的 ID。

```
aws cloudfront get-distribution-config --id primary_distribution_ID --output
 yaml > primary-distribution.yaml
```

2. 打开刚创建的名为 `primary-distribution.yaml` 的文件。编辑文件，进行以下更改：

   - 将持续部署策略 ID（从上一步中复制的）粘贴到 `ContinuousDeploymentPolicyId` 字段中。
   - 将 `ETag` 字段重命名为 `IfMatch`，但不更改字段的值。

   完成后保存该文件。

3. 使用以下命令更新主分配以使用持续部署策略。将 *`primary_distribution_ID`* 替换为主分配的 ID。

```
aws cloudfront update-distribution --id primary_distribution_ID --cli-input-yaml
 file://primary-distribution.yaml
```

当主分配的配置以及附加的持续部署策略部署到边缘站点时，CloudFront 会开始根据流量配置将指定部分的流量发送到暂存分配。

API

要使用 CloudFront API 创建暂存分配和持续部署策略，请使用以下 API 操作：

- CopyDistribution
- CreateContinuousDeploymentPolicy

有关您在这些 API 调用中指定的字段的更多信息，请参阅以下内容：

- the section called “将请求路由到暂存分配”
- the section called “基于权重的配置的会话粘性”
- 有关 AWS SDK 或其他 API 客户端的 API 参考文档

创建暂存分配和持续部署策略后，使用 UpdateDistribution（在主分配上）将持续部署策略附加到主分配。

更新暂存分配

以下步骤演示如何使用持续部署策略更新暂存分配。

Console

您可以更新主分配和暂存分配的某些配置。有关更多信息，请参阅 更新主分配和暂存分配。

更新暂存分配（控制台）

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 在导航窗格中，选择分配。
3. 选择主分配。这是当前为生产流量提供服务的分配，您将从中创建暂存分配。
4. 选择查看暂存分配。
5. 使用控制台修改暂存分配的配置。有关您可以更新的设置的更多信息，请参阅the section called “更新主分配和暂存分配”。

一旦暂存分配的配置部署到边缘站点，它就会对路由到暂存分配的传入流量生效。

CLI

更新暂存分配（带输入文件的 CLI）

1. 使用以下命令将暂存分配的配置保存到一个名为 `staging-distribution.yaml` 的文件中。将 *`staging_distribution_ID`* 替换为暂存分配的 ID。

```
aws cloudfront get-distribution-config --id staging_distribution_ID --output
 yaml > staging-distribution.yaml
```

2. 打开刚创建的名为 `staging-distribution.yaml` 的文件。编辑文件，进行以下更改：

- 修改暂存分配的配置。有关您可以更新的设置的更多信息，请参阅the section called “更新主分配和暂存分配”。
- 将 `ETag` 字段重命名为 `IfMatch`，但不更改字段的值。

完成后保存该文件。

3. 使用以下命令更新暂存分配的配置。将 *`staging_distribution_ID`* 替换为暂存分配的 ID。

```
aws cloudfront update-distribution --id staging_distribution_ID --cli-input-yaml
 file://staging-distribution.yaml
```

一旦暂存分配的配置部署到边缘站点，它就会对路由到暂存分配的传入流量生效。

API

要更新暂存分配的配置，请使用 UpdateDistribution（在暂存分配上）修改暂存分配的配置。有关您可以更新的设置的更多信息，请参阅the section called “更新主分配和暂存分配”。

更新持续部署策略

以下步骤演示如何更新持续部署策略。

Console

您可以通过更新持续部署策略来更新分配的流量配置。

更新持续部署策略（控制台）

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 在导航窗格中，选择分配。
3. 选择主分配。这是当前为生产流量提供服务的分发，您将从中创建暂存分发。
4. 在持续部署部分中，选择编辑策略。
5. 修改持续部署策略中的流量配置。在完成后，选择保存更改。

当主分配的配置以及更新的持续部署策略部署到边缘站点时，CloudFront 会开始根据更新的流量配置将流量发送到暂存分配。

CLI

更新持续部署策略（带输入文件的 CLI）

1. 使用以下命令将持续部署策略的配置保存到一个名为 `continuous-deployment-policy.yaml` 的文件中。将 *`continuous_deployment_policy_ID`* 替换为持续部署策略的 ID。以下命令使用转义字符（\）和换行符来提高可读性，但您应在该命令中省略这些字符。

```
aws cloudfront get-continuous-deployment-policy-config --
id continuous_deployment_policy_ID \
                                                    --output yaml >
 continuous-deployment-policy.yaml
```

2. 打开刚创建的名为 `continuous-deployment-policy.yaml` 的文件。编辑文件，进行以下更改：

   - 根据需要修改持续部署策略的配置。例如，您可以从使用基于标头的流量配置更改为使用基于权重的流量配置，也可以更改基于权重的配置的流量百分比（权重）。有关更多信息，请参阅the section called “将请求路由到暂存分配” 和the section called “基于权重的配置的会话粘性”。
   - 将 `ETag` 字段重命名为 `IfMatch`，但不更改字段的值。

   完成后保存该文件。

3. 使用以下命令更新持续部署策略。将 *`continuous_deployment_policy_ID`* 替换为持续部署策略的 ID。以下命令使用转义字符（\）和换行符来提高可读性，但您应在该命令中省略这些字符。

```
aws cloudfront update-continuous-deployment-policy --
id continuous_deployment_policy_ID \
                                                --cli-input-yaml file://
continuous-deployment-policy.yaml
```

当主分配的配置以及更新的持续部署策略部署到边缘站点时，CloudFront 会开始根据更新的流量配置将流量发送到暂存分配。

API

要更新持续部署策略，请使用 UpdateContinuousDeploymentPolicy

提升暂存分配的配置

以下步骤演示如何提升暂存分配的配置。

Console

在提升暂存分配时，CloudFront 会将配置从暂存分配复制到主分配。CloudFront 还将禁用持续部署策略，并将所有流量路由到主分配。

提升配置后，可以在下次要测试配置更改时重用相同的暂存分配。

提升暂存分配的配置（控制台）

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 在导航窗格中，选择分配。
3. 选择主分配。这是当前为生产流量提供服务的分发，您将从中创建暂存分发。
4. 在持续部署部分中，选择提升。
5. 键入 **`confirm`**，然后选择提升。

CLI

在提升暂存分配时，CloudFront 会将配置从暂存分配复制到主分配。CloudFront 还将禁用持续部署策略，并将所有流量路由到主分配。

提升配置后，可以在下次要测试配置更改时重用相同的暂存分配。

提升暂存分配的配置（CLI）

- 使用 aws cloudfront update-distribution-with-staging-config 命令将暂存分配的配置提升为主分配。以下示例命令使用转义字符（\）和换行符来提高可读性，但您应在该命令中省略这些字符。在以下示例命令中：
  - 将 *`primary_distribution_ID`* 替换为主分配的 ID。

- 将 *staging_distribution_ID* 替换为暂存分配的 ID。
- 将 *primary_distribution_ETag* 和 *staging_distribution_ETag* 分别替换为主分配和暂存分配的 ETag 值。确保主分配的值是第一个，如示例所示。

```
aws cloudfront update-distribution-with-staging-config --
id primary_distribution_ID \
                                                    --staging-distribution-
id staging_distribution_ID \
                                                    --if-match
 'primary_distribution_ETag,staging_distribution_ETag'
```

API

要将暂存分配的配置提升为主分配，请使用 UpdateDistributionWithStagingConfig。

监控暂存分配

要监控暂存分配的性能，您可以使用 CloudFront 为所有分发提供的相同的指标、日志和报告。例如：

- 您可以在 CloudFront 控制台中查看默认 CloudFront 分配指标（例如，总请求数和错误率），也可以启用其他指标（例如，按状态码划分的缓存命中率和错误率），但这需要支付额外费用。您也可以根据这些指标创建警报。
- 您可以查看标准日志和实时日志，以获取有关暂存分配收到的请求的详细信息。标准日志包含以下两个字段，可帮助您识别在 CloudFront 将请求路由到暂存分配之前最初将请求发送到的主分配：`primary-distribution-id` 和 `primary-distribution-dns-name`。
- 您可以在 CloudFront 控制台中查看和下载报告，例如缓存统计数据报告。

了解持续部署的工作方式

以下主题说明了 CloudFront 持续部署的工作方式。

主题

- 将请求路由到暂存分配
- 基于权重的配置的会话粘性

- 更新主分配和暂存分配
- 主分配和暂存分配不共享缓存

将请求路由到暂存分配

在使用 CloudFront 持续部署时，您无需更改有关查看器请求的任何内容。查看器无法使用 DNS 名称、IP 地址或 CNAME 将请求直接发送到暂存分配。相反，查看器将请求发送到主（生产）分发，CloudFront 会根据持续部署策略中的流量配置设置将其中一些请求路由到暂存分配。有两种类型的流量配置：

基于权重

基于权重的配置将指定百分比的查看器请求路由到暂存分配。在使用基于权重的配置时，您也可以启用会话粘性，这有助于确保 CloudFront 将来自同一查看器的请求视为单个会话的一部分。有关更多信息，请参阅 the section called “基于权重的配置的会话粘性”。

基于标头

当查看器请求包含特定的 HTTP 标头（您可以指定标头和值）时，基于标头的配置会将请求路由到暂存分配。不包含指定的标头和值的请求将路由到主分配。在本地测试中或在您能够控制查看器请求时，此配置很有用。

> **Note**
>
> 路由到暂存分配的标头必须包含前缀 `aws-cf-cd-`。

基于权重的配置的会话粘性

在使用基于权重的配置将流量路由到暂存分配时，您也可以启用会话粘性，这有助于确保 CloudFront 将来自同一查看器的请求视为单个会话的一部分。在启用会话粘性时，CloudFront 会设置 Cookie，以便单个会话中的来自同一查看器的所有请求都由一个分发（主分配或暂存分配）提供服务。

启用会话粘性时，还可以指定空闲持续时间。如果查看器在这段时间内处于空闲状态（未发送任何请求），则会话将过期，并且 CloudFront 会将来自该查看器的将来请求视为一个新会话。您可以将空闲持续时间指定为秒数，从 300（五分钟）到 3600（一小时）。

在以下情况下，CloudFront 将重置所有会话（甚至是活动会话）并将所有请求视为新会话：

- 您禁用或启用持续部署策略

- 您禁用或启用会话粘性设置

更新主分配和暂存分配

当主分配附加了持续部署策略时，以下配置更改适用于主分配和暂存分配：

- 所有缓存行为设置，包括默认缓存行为
- 所有源设置（源和源组）
- 自定义错误响应（错误页面）
- 地理限制
- 默认根对象
- 日志记录设置
- 描述（评论）

您还可以更新分发的配置中引用的外部资源，例如缓存策略、响应标头策略、CloudFront 函数或 Lambda@Edge 函数。

主分配和暂存分配不共享缓存

主分配和暂存分配不共享缓存。当 CloudFront 向暂存分配发送第一个请求时，其缓存为空。当请求到达暂存分配时，它会开始缓存响应（如果已配置为执行此操作）。

持续部署的配额和其他注意事项

CloudFront 持续部署受以下配额和其他注意事项的约束。

配额

- 每个 AWS 账户 的最大暂存分配数：20
- 每个 AWS 账户 的最大持续部署策略数：20
- 在基于权重的配置中，您可以发送到暂存分配的流量的最大百分比：15%
- 会话粘性空闲持续时间的最小和最大值：300–3600 秒

有关更多信息，请参阅 配额。

> **Note**
>
> 使用持续部署并且您的主分配设置为 OAC 以进行 S3 存储桶访问时，请更新您的 S3 存储桶策略以允许访问暂存分配。有关 S3 存储桶策略示例，请参阅the section called “向源访问控制授予访问 S3 存储桶的权限”。

AWS WAF Web ACL

如果您为分配启用持续分配，那么以下注意事项适用于 AWS WAF：

- 您无法第一次就将 AWS WAF Web 访问控制列表（ACL）关联到分配。
- 您无法解除 AWS WAF Web ACL 与分配的关联。

在执行上述任务之前，您必须删除生产分配的持续部署策略。这也会删除暂存分配。有关更多信息，请参阅 使用 AWS WAF 保护功能。

CloudFront 将所有请求发送到主分配的情况

在某些情况下（例如，资源利用率较高的时段），无论持续部署策略中指定的内容如何，CloudFront 都可能将所有请求发送到主分配。

无论持续部署策略中指定的内容如何，CloudFront 都会在流量高峰时段将所有请求发送到主分配。峰值流量是指 CloudFront 服务上的流量，而不是您的分配上的流量。

HTTP/3

您不能将持续部署与支持 HTTP/3 的分发结合使用。

在 CloudFront 分配中使用各种源

当您创建分配时，可指定 CloudFront 在其中发送对于文件的请求的源。您可以在 CloudFront 中使用多种不同的源。例如，您可以使用 Amazon S3 存储桶、MediaStore 容器、MediaPackage 通道、Application Load Balancer 或 AWS Lambda 函数 URL。

主题

- 使用 Amazon S3 存储桶
- 使用 MediaStore 容器或 MediaPackage 通道

- 使用应用程序负载均衡器
- 使用 Lambda 函数 URL
- 使用 Amazon EC2（或其他自定义源）
- 使用 CloudFront 源组

使用 Amazon S3 存储桶

以下主题介绍您可以使用 Amazon S3 存储桶作为 CloudFront 分配的源的不同方法。

主题

- 使用标准 Amazon S3 存储桶
- 使用 Amazon S3 对象 Lambda
- 使用 Amazon S3 接入点
- 使用配置为网站端点的 Amazon S3 存储桶
- 将 CloudFront 添加到现有 Amazon S3 存储桶
- 将 Amazon S3 存储桶移至其他 AWS 区域

使用标准 Amazon S3 存储桶

当您使用 Amazon S3 作为分配的源时，可将希望 CloudFront 传送的对象放在 Amazon S3 存储桶中。您可以使用 Amazon S3 支持的任何方法将对象放入 Amazon S3。例如，您可以使用 Amazon S3 控制台或 API 或第三方工具。您可在存储桶中创建一个层次结构来存储对象，就如您使用任何其他标准 Amazon S3 存储桶一样。

使用现有 Amazon S3 存储桶作为您的 CloudFront 源服务器不会以任何方式改变存储桶；您仍可如同往常一样使用它来存储和访问 Amazon S3 对象（标准 Amazon S3 价格）。在存储桶中存储对象会产生常规的 Amazon S3 费用。有关使用 CloudFront 的费用的更多信息，请参阅 Amazon CloudFront 定价。有关将 CloudFront 与现有 S3 存储桶结合使用的更多信息，请参阅the section called “将 CloudFront 添加到现有 Amazon S3 存储桶”。

> **Important**
>
> 要使存储桶能够用于 CloudFront，其名称必须符合 DNS 命名要求。有关更多信息，请转至《Amazon Simple Storage Service 用户指南》中的存储桶命名规则。

在指定 Amazon S3 存储桶作为 CloudFront 的源时，建议您使用以下格式：

bucket-name.s3.*region*.amazonaws.com

当您以该格式指定存储桶名称时，可以使用以下 CloudFront 功能：

- 将 CloudFront 配置为使用 SSL/TLS 与您的 Amazon S3 存储桶通信。有关更多信息，请参阅 the section called “将 HTTPS 与 CloudFront 结合使用”。
- 使用源访问控制要求查看器使用 CloudFront URL（而非使用 Amazon S3 URL）访问您的内容。有关更多信息，请参阅 the section called “限制对 Amazon Simple Storage Service 源的访问”。
- 通过向 CloudFront 提交 POST 和 PUT 请求，更新您的存储桶内容。有关更多信息，请参阅 the section called “HTTP 方法”主题中的 the section called “CloudFront 如何处理请求并将请求转发到您的 Amazon S3 源”。

切勿使用以下格式指定存储桶：

- Amazon S3 路径样式：s3.amazonaws.com/*bucket-name*
- Amazon S3 CNAME

使用 Amazon S3 对象 Lambda

当您创建对象 Lambda 接入点时，Amazon S3 会自动为您的对象 Lambda 接入点生成一个唯一的别名。您可以使用此别名代替 Amazon S3 存储桶名称作为您 CloudFront 分配的源。

当您使用对象 Lambda 接入点别名作为 CloudFront 的源时，建议您使用以下格式：

alias.s3.*region*.amazonaws.com

有关查找 *alias* 的详细信息，请参阅《Amazon S3 用户指南》中的如何为您的 S3 存储桶对象 Lambda 接入点使用存储桶式别名。

> **Important**
>
> 当您使用对象 Lambda 接入点作为 CloudFront 的源时，必须使用源访问控制。

有关示例使用案例，请参阅将 Amazon S3 对象 Lambda 与 Amazon CloudFront 配合使用，以为最终用户量身定制内容。

CloudFront 将对象 Lambda 接入点源视为标准 Amazon S3 存储桶源。

如果使用 Amazon S3 对象 Lambda 作为您分配的源，则必须配置以下四种权限。

Object Lambda Access Point

为对象 Lambda 接入点添加权限

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，选择对象 Lambda 接入点。
3. 请选择要使用的对象 Lambda 接入点。
4. 选择权限选项卡。
5. 在对象 Lambda 接入点策略部分中选择编辑。
6. 将以下策略粘贴到策略字段中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3-object-lambda:Get*",
            "Resource": "arn:aws:s3-object-lambda:region:AWS-account-
ID:accesspoint/Object-Lambda-Access-Point-name",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:cloudfront::AWS-account-
ID:distribution/CloudFront-distribution-ID"
                }
            }
        }
    ]
}
```

7. 选择保存更改。

Amazon S3 Access Point

为 Amazon S3 接入点添加权限

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，选择接入点。
3. 请选择要使用的 Amazon S3 接入点。
4. 选择权限选项卡。
5. 在接入点策略部分中选择编辑。
6. 将以下策略粘贴到策略字段中。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "s3objlambda",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:region:AWS-account-ID:accesspoint/Access-Point-
name",
                "arn:aws:s3:region:AWS-account-ID:accesspoint/Access-Point-name/
object/*"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": "s3-object-lambda.amazonaws.com"
                }
            }
        }
    ]
}
```

7. 选择保存。

Amazon S3 bucket

添加权限至 Amazon S3 存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，选择桶。
3. 请选择要使用的 Amazon S3 存储桶。
4. 选择权限选项卡。
5. 在存储桶策略部分，选择编辑。
6. 将以下策略粘贴到策略字段中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "*",
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:DataAccessPointAccount": "AWS-account-ID"
                }
            }
        }
    ]
}
```

7. 选择保存更改。

AWS Lambda function

向 Lambda 函数添加权限

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 在导航窗格中，选择函数。
3. 选择要使用的 AWS Lambda 函数。
4. 选择配置选项卡，然后选择权限。
5. 在基于资源的策略声明部分中选择添加权限。
6. 选择AWS 账户。
7. 输入声明 ID 的名称。
8. 对于委托人，请输入 `cloudfront.amazonaws.com`。
9. 从操作下拉菜单中选择 `lambda:InvokeFunction`。
10. 选择保存。

使用 Amazon S3 接入点

当您使用 S3 接入点时，Amazon S3 会自动为您生成一个唯一的别名。您可以使用此别名代替 Amazon S3 存储桶名称作为您 CloudFront 分配的源。

当您使用 Amazon S3 接入点别名作为 CloudFront 的源时，建议您使用以下格式：

alias.s3.*region*.amazonaws.com

有关如何查找 *alias* 的详细信息，请参阅《Amazon S3 用户指南》中的对您的 S3 存储桶接入点使用存储桶式别名。

> **Important**
>
> 当您使用 Amazon S3 接入点作为 CloudFront 的源时，必须使用源访问控制。

CloudFront 将 Amazon S3 接入点源视为与标准 Amazon S3 存储桶源相同。

如果使用 Amazon S3 对象 Lambda 作为您分配的源，则必须配置以下两种权限。

Amazon S3 Access Point

为 Amazon S3 接入点添加权限

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，选择接入点。
3. 请选择要使用的 Amazon S3 接入点。
4. 选择权限选项卡。
5. 在接入点策略部分中选择编辑。
6. 将以下策略粘贴到策略字段中。

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "s3objlambda",
            "Effect": "Allow",
            "Principal": {"Service": "cloudfront.amazonaws.com"},
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:region:AWS-account-ID:accesspoint/Access-Point-
name",
                "arn:aws:s3:region:AWS-account-ID:accesspoint/Access-Point-name/
object/*"
            ],
            "Condition": {
                "StringEquals": {"aws:SourceArn": "arn:aws:cloudfront::AWS-
account-ID:distribution/CloudFront-distribution-ID"}
            }
        }
    ]
}
```

7. 选择保存。

Amazon S3 bucket

添加权限至 Amazon S3 存储桶

1. 登录到AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在导航窗格中，选择桶。
3. 请选择要使用的 Amazon S3 存储桶。
4. 选择权限选项卡。
5. 在存储桶策略部分，选择编辑。
6. 将以下策略粘贴到策略字段中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "*",
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:DataAccessPointAccount": "AWS-account-ID"
                }
            }
        }
    ]
}
```

7. 选择保存更改。

使用配置为网站端点的 Amazon S3 存储桶

您可以使用配置为网站端点的 Amazon S3 存储桶作为 CloudFront 的自定义源。当您配置 CloudFront 分配时，对于源，请输入您的存储桶的 Amazon S3 静态网站托管端点。此值显示在 Amazon S3 控制台的属性选项卡上的静态网站托管窗格中。例如：

```
http://bucket-name.s3-website-region.amazonaws.com
```

有关指定 Amazon S3 静态网站端点的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的网站端点。

当您以此格式将存储桶名称指定为您的源时，可使用 Amazon S3 重定向和 Amazon S3 自定义错误文档。有关更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的配置自定义错误文档和配置重新导向。（CloudFront 也提供自定义错误页面。有关更多信息，请参阅the section called “为特定 HTTP 状态代码创建自定义错误页面”。）

使用 Amazon S3 存储桶作为您的 CloudFront 源服务器不会以任何方式更改存储桶。您仍可像往常一样使用它，并且会产生定期 Amazon S3 费用。有关使用 CloudFront 的费用的更多信息，请参阅 Amazon CloudFront 定价。

> **Note**
>
> 如果您使用 CloudFront API 创建包含配置为网站端点的 Amazon S3 存储桶的分配，则必须使用 `CustomOriginConfig` 对该分配进行配置，即使该网站是在 Amazon S3 存储桶中托管。有关使用 CloudFront API 创建分配的更多信息，请参阅《Amazon CloudFront API 参考》中的 CreateDistribution。

将 CloudFront 添加到现有 Amazon S3 存储桶

如果您将对象存储在 Amazon S3 存储桶中，您可以让用户直接从 S3 中获取这些对象，也可以将 CloudFront 配置为从 S3 中获取这些对象并将其分发到用户。如果您的用户频繁访问您的对象（因为在使用量较大的情况下，CloudFront 数据传输价格低于 Amazon S3 数据传输价格），则使用 CloudFront 可能会更经济高效。此外，使用 CloudFront 时下载速度也比仅使用 Amazon S3 时更快，因为您的对象存储在离您用户更近的位置。

> **Note**
>
> 如果您希望 CloudFront 尊重 Amazon S3 跨源资源共享设置，请将 CloudFront 配置为将 `Origin` 标头转发到 Amazon S3。有关更多信息，请参阅 the section called “根据请求标头缓存内容”。

如果您当前使用自己的域名（如 example.com）直接从您的 Amazon S3 存储桶中分发内容，而不是使用 Amazon S3 存储桶的域名（如 DOC-EXAMPLE-BUCKET.s3.us-west-2.amazonaws.com），则可以按照以下过程添加 CloudFront 而不会造成中断。

在您已经从 Amazon S3 分发内容时添加 CloudFront

1. 创建 CloudFront 分配。有关更多信息，请参阅 the section called “创建分配”。

   当您创建分配时，请指定您的 Amazon S3 存储桶名称作为源服务器。

   > **Important**
   >
   > 要使存储桶能够用于 CloudFront，其名称必须符合 DNS 命名要求。有关更多信息，请转至《Amazon Simple Storage Service 用户指南》中的存储桶命名规则。

   如果您对 Amazon S3 使用 CNAME，请也为您的分配指定 CNAME。
2. 创建一个测试网页并在该网页中添加您的 Amazon S3 存储桶中公开可读对象的链接，然后测试这些链接。对于此初步测试，请在对象 URL 中使用此分配的 CloudFront 域名，例如 `https://d111111abcdef8.cloudfront.net/images/image.jpg`。

   有关 CloudFront URL 格式的更多信息，请参阅the section called “自定义文件 URL”。
3. 如果使用 Amazon S3 CNAME，应用程序会使用域名（例如，example.com）来引用 Amazon S3 存储桶中的对象，而不是使用存储桶名称（例如，DOC-EXAMPLE-BUCKET.s3.amazonaws.com）。要继续使用您的域名来引用对象，而不是使用分配的 CloudFront 域名（例如，d111111abcdef8.cloudfront.net），您需要通过您的 DNS 服务提供商更新您的设置。

   要使 Amazon S3 CNAME 发挥作用，您的 DNS 服务提供商必须具有针对您的域的 CNAME 资源记录集，当前将对域的查询路由到您的 Amazon S3 存储桶。例如，如果用户请求该对象：

   ```
   https://example.com/images/image.jpg
   ```

该请求将自动重新路由，并且用户将看到该对象：

```
https://DOC-EXAMPLE-BUCKET.s3.amazonaws.com/images/image.jpg
```

要将查询按指定路线发送到您的 CloudFront 分配而不是您的 Amazon S3 存储桶，您需要使用 DNS 服务提供商提供的方法来为您的域更新 CNAME 资源记录集。此更新后的别名记录会将 DNS 查询从您的域重新导向到分配的 CloudFront 域名。有关更多信息，请参阅 DNS 服务提供商提供的文档。

> **Note**
>
> 如果您使用 Route 53 作为您的 DNS 服务，则可使用 CNAME 资源记录集或别名资源记录集。有关编辑资源记录集的信息，请参阅编辑记录。有关别名资源记录集的信息，请参阅在别名和非别名记录之间进行选择。这两个主题都位于《Amazon Route 53 开发人员指南》中。

有关对 CloudFront 使用 CNAME 的更多信息，请参阅the section called “使用自定义 URL”。

在您更新 CNAME 资源记录集后，将更改传播到整个 DNS 系统可能需要长达 72 个小时的时间，虽然通常传播的速度较快。在此期间，会继续将您的内容的一些请求按指定路线发送到您的 Amazon S3 存储桶，而其他请求将被发送到 CloudFront。

将 Amazon S3 存储桶移至其他 AWS 区域

如果您使用 Amazon S3 作为 CloudFront 分配的源，并且您将存储桶移至其他 AWS 区域，则在满足下面两个条件的情况下，CloudFront 可能要花费多达 1 个小时的时间来更新其记录以使用新的区域：

- 您使用 CloudFront 源访问身份 (OAI) 限制对存储桶的访问。
- 您将存储桶移至需要签名版 4 以进行身份验证的 Amazon S3 区域。

在使用 OAI 时，CloudFront 使用区域（以及其他值）来计算用于从存储桶中请求对象的签名。有关 OAI 的更多信息，请参阅the section called “使用源访问身份（旧版，不推荐）”。有关支持签名版本 2 的 AWS 区域的列表，请参阅《Amazon Web Services 一般参考》中的签名版本 2 签名流程。

要强制更快地更新 CloudFront 的记录，您可以通过更新 CloudFront 控制台中常规选项卡上的说明字段来更新您的 CloudFront 分配。在更新分配时，CloudFront 会立即检查您的存储桶所在的区域。将更改传播到所有边缘站点应该只需几分钟时间。

使用 MediaStore 容器或 MediaPackage 通道

要使用 CloudFront 流式传输视频，您可以设置配置为 MediaStore 容器的 Amazon S3 存储桶，或使用 MediaPackage 创建通道和端点。然后，在 CloudFront 中创建并配置分配以流式传输视频。

有关更多信息和分步说明，请参阅以下主题：

- the section called “使用 AWS Elemental MediaStore 作为源来提供视频”
- the section called “提供使用 AWS Elemental MediaPackage 格式化的实时视频”

使用应用程序负载均衡器

如果您的源是托管在一个或多个 Amazon EC2 实例上的一个或多个 HTTP(S) 服务器（Web 服务器），则可以使用面向互联网的应用程序负载均衡器向实例分配流量。面向互联网的负载均衡器具有可公开解析的 DNS 名称，并通过互联网将来自客户端的请求路由到目标。

有关使用应用程序负载均衡器作为 CloudFront 的源的更多信息，包括如何确保查看器只能通过 CloudFront 而不是直接访问负载均衡器来访问您的 Web 服务器，请参阅the section called “限制访问应用程序负载均衡器”。

使用 Lambda 函数 URL

Lambda 函数 URL 是 Lambda 函数的专用 HTTPS 端点。您可以使用 Lambda 函数 URL 完全在 Lambda 中构建无服务器 Web 应用程序。您可以直接通过函数 URL 调用 Lambda Web 应用程序，而无需与 API Gateway 或应用程序负载均衡器集成。

如果您使用带有函数 URL 的 Lambda 函数构建无服务器 Web 应用程序，则可以添加 CloudFront 以获得以下好处：

- 通过将内容缓存在更靠近查看器的位置来加速应用程序
- 对您的 Web 应用程序使用自定义域
- 使用 CloudFront 缓存行为将不同的 URL 路径路由到不同的 Lambda 函数
- 使用 CloudFront 地理限制或 AWS WAF（或两者）阻止特定请求

- 将 AWS WAF 与 CloudFront 结合使用，以帮助保护您的应用程序免受恶意自动程序侵害，帮助防止常见的应用程序漏洞，并增强对 DDoS 攻击的防范

要使用 Lambda 函数 URL 作为 CloudFront 分配的源，请将 Lambda 函数 URL 的完整域名指定为源域。Lambda 函数 URL 域名使用以下格式：

function-URL-ID.lambda-url.*AWS-Region*.on.aws

当您使用 Lambda 函数 URL 作为 CloudFront 分配的源时，函数 URL 必须可供公开访问。为此，请使用以下选项之一：

- 如果您使用源访问控制（OAC）功能，Lambda 函数 URL 的 AuthType 参数必须使用 AWS_IAM 值并在基于资源的策略中允许 lambda:InvokeFunctionUrl 权限。有关将 Lambda 函数 URL 用于 OAC 的更多信息，请参阅限制对 AWS Lambda 函数 URL 源的访问。
- 如果您不使用 OAC，可以将函数 URL 的 AuthType 参数设置为 NONE，并在基于资源的策略中允许 lambda:InvokeFunctionUrl 权限。

您也可以在 CloudFront 发送到源的请求中添加自定义源标头。如果请求中不存在此标头，则编写函数代码以返回错误响应。这有助于确保用户只能通过 CloudFront（而无法直接使用 Lambda 函数 URL）来访问您的 Web 应用程序。

有关 Lambda 函数 URL 的更多信息，请参阅《AWS Lambda 开发人员指南》中的以下主题：

- Lambda 函数 URL – Lambda 函数 URL 功能的一般概览
- 调用 Lambda 函数 URL – 包括有关用于对无服务器 Web 应用程序进行编码的请求和响应有效负载的详细信息
- Lambda 函数 URL 的安全性和身份验证模型 - 包括有关 Lambda 身份验证类型的详细信息

使用 Amazon EC2（或其他自定义源）

自定义源是具有可公开解析的 DNS 名称的 HTTP(S) Web 服务器，可通过互联网将来自客户端的请求路由到目标。HTTP(S) 服务器可以托管在 AWS（例如 Amazon EC2 实例）上，也可以托管在其他地方。配置为网站端点的 Amazon S3 源也被视为自定义源。有关更多信息，请参阅 the section called “使用配置为网站端点的 Amazon S3 存储桶”。

当您使用自己的 HTTP 服务器作为自定义源时，可指定服务器的 DNS 名称、HTTP 和 HTTPS 端口以及您在从源中提取对象时希望 CloudFront 使用的协议。

当您使用自定义源时，大部分 CloudFront 功能都受支持，但私有内容除外。虽然您可以使用签名 URL 从自定义源分配内容，但要让 CloudFront 访问自定义源，该源必须保持可公共访问。有关更多信息，请参阅 the section called “使用签名 URL 和签名 Cookie 限制内容”。

请遵照将 Amazon EC2 实例和其他自定义源与 CloudFront 结合使用的准则。

- 在为同一 CloudFront 源提供内容的所有服务器上托管和提供相同的内容。有关更多信息，请参阅the section called “源设置”主题中的the section called “分配设置”。
- 如果需要 AWS Support 或 CloudFront 使用此值进行调试，请在所有服务器上记录 `X-Amz-Cf-Id` 标头条目。
- 限制对自定义源所侦听的 HTTP 和 HTTPS 端口的请求。
- 在您实现的过程中，使所有服务器的时钟同步。请注意，CloudFront 对签名 URL 和签名 Cookie、日志和报告使用协调世界时 (UTC)。此外，如果您使用 CloudWatch 指标监控 CloudFront 活动，请注意，CloudWatch 也使用 UTC。
- 使用冗余服务器来处理故障。
- 有关使用自定义源来提供私有内容的信息，请参阅the section called “在自定义源上限制对文件的访问”。
- 有关请求和响应行为以及受支持的 HTTP 状态代码的信息，请参阅 请求和响应行为。

如果您为自定义源使用 Amazon EC2，建议您执行下列操作：

- 使用亚马逊机器映像为 Web 服务器自动安装软件。有关的更多信息，请参阅 Amazon EC2 文档。
- 使用 Elastic Load Balancing 负载平衡器来处理通过多个 Amazon EC2 实例的流量，并使您的应用程序与 Amazon EC2 实例的变更隔离。例如，如果您使用负载平衡器，则可添加和删除 Amazon EC2 实例，而无需更改您的应用程序。有关更多信息，请参阅 Elastic Load Balancing 文档。
- 当您创建 CloudFront 分配时，请为源服务器的域名指定负载平衡器的 URL。有关更多信息，请参阅 the section called “创建分配”。

使用 CloudFront 源组

您可以为您的 CloudFront 源指定源组，例如，在您希望为需要高可用性的场景配置源故障转移的情况下。使用源故障转移为 CloudFront 指定主源以及 CloudFront 将在主源返回特定 HTTP 状态代码故障响应时自动切换到的次要源。

有关更多信息，包括设置源组的步骤，请参阅the section called “使用源失效转移来提高可用性”。

通过添加备用域名（CNAME）使用自定义 URL

在创建分配时，CloudFront 会为其提供域名，例如 d111111abcdef8.cloudfront.net。您可以使用备用域名（也称为 CNAME），而不是使用提供的此域名。

要了解如何使用您自己的域名（例如 www.example.com），请参阅以下主题：

主题

使用备用域名的要求

在向 CloudFront 分配添加备用域名（例如 www.example.com）时，需满足以下要求：

备用域名必须小写

所有备用域名 (CNAME) 都必须小写。

备用域名必须为有效的 SSL/TLS 证书所涵盖

要将备用域名 (CNAME) 添加至 CloudFront 分配，必须给您的分配附加一个可信且有效的 SSL/TLS 证书，其中涵盖了备用域名。这样可以确保仅有权访问您的域证书的人员能够将 CloudFront 关联到与您的域相关的 CNAME。

受信任证书是由 AWS Certificate Manager（ACM）或其他有效证书颁发机构（CA）颁发的证书。您可以使用自签名证书来验证现有的 CNAME，但不能 验证新的 CNAME。CloudFront 支持与 Mozilla 相同的证书颁发机构。有关当前列表，请参阅 Mozilla 包含的 CA 证书列表。

为了使用附加的证书来验证备用域名（包括含通配符的域名），CloudFront 会检查证书上的使用者备用名称 (SAN)。要添加的备用域名必须为 SAN 所涵盖。

> **Note**
>
> 一次只能将一个证书附加到 CloudFront 分配。

您可以通过执行以下操作之一来证明您有权向分配中添加特定的备用域名：

- 附加包含备用域名的证书，例如 product-name.example.com。
- 附加一个证书，其中包括一个 * 通配符在域名的开头，以用一个证书涵盖多个子域。当指定通配符时，可以在 CloudFront 中添加多个子域作为备用域名。

以下示例说明了如何在证书的域名中使用通配符来授权您在 CloudFront 中添加特定的备用域名。

- 您要添加 marketing.example.com 作为备用域名。您在证书中列示以下域名：*.example.com。当您将此证书附加到 CloudFront 时，可以为您的分配添加任何备用域名，以替换该层次上的通配符，包括 marketing.example.com。例如，您还可以添加以下备用域名：
  - product.example.com
  - api.example.com

  但是，不能添加所在层次高于或低于通配符的备用域名。例如，您不能添加备用域名 example.com 或 marketing.product.example.com。
- 您要添加 example.com 作为备用域名。要执行此操作，必须在附加到分配的证书上列出域名 example.com 本身。
- 您要添加 marketing.product.example.com 作为备用域名。要执行此操作，您可以在证书上列出 *.product.example.com，也可以在证书上列出 marketing.product.example.com 本身。

更改 DNS 配置所需的权限

在添加备用域名时，您必须创建 CNAME 记录，以将备用域名的 DNS 查询路由到您的 CloudFront 分配。要执行此操作，您必须具有创建 CNAME 记录的权限，并为您使用的备用域名指定 DNS 服务提供商。通常，这意味着您拥有这些域，但您可能在为域所有者开发应用程序。

备用域名和 HTTPS

如果您希望查看器使用 HTTPS 和备用域名，还必须进行其他配置。有关更多信息，请参阅 使用备用域名和 HTTPS。

备用域名的使用限制

请注意备用域名的以下使用限制：

备用域名的最大数量

有关您可以添加到分配的当前最大备用域名数，或者要请求提高配额（以前称为限制），请参阅分配的一般配额。

重复和重合备用域名

如果相同的备用域名已经在另一个 CloudFront 分配中存在，则您不能将其添加到 CloudFront 分配中，即使您的 AWS 账户拥有其他分配也不行。

不过，您可以添加通配符备用域名（例如 *.example.com），该域名包含（重叠）非通配符备用域名，例如 www.example.com。如果在两个分配中有重合的备用域名，则 CloudFront 将发送请求到具有更具体名称匹配的那个分配，而不管 DNS 记录指向哪个分配。例如，marketing.domain.com 比 *.domain.com 更具体。

域前置

CloudFront 中包含防止域前置跨不同 AWS 账户发生的措施。在域前置中，非标准客户端会建立与一个 AWS 账户中的域名的 TLS/SSL 连接，但随后又发出针对另一个 AWS 账户中的不相关名称的 HTTPS 请求。例如，TLS 连接可能连接到 www.example.com，然后发出对 www.example.org 的 HTTP 请求。

为防范跨不同 AWS 账户的域前置攻击，CloudFront 确保拥有为特定连接提供证书的 AWS 账户始终与拥有在同一连接上被处理的请求的 AWS 账户匹配。

如果两个 AWS 账号不匹配，CloudFront 将做出“HTTP 421 错误定向请求”响应，让客户端有机会使用正确的域进行连接。

在域的顶部节点（顶级域名）添加备用域名

当您将备用域名添加到分配时，通常在 DNS 配置中创建 CNAME 记录，以将域名的 DNS 查询路由到 CloudFront 分配。不过，您无法为 DNS 命名空间的顶端节点（也称为顶级域名）创建 CNAME 记录；DNS 协议不允许您这样做。例如，如果您注册了 DNS 名称 example.com，则顶级域名为 example.com。您不能为 example.com 创建 CNAME 记录，但可以为 www.example.com、newproduct.example.com 等创建 CNAME 记录。

如果您使用 Route 53 作为您的 DNS 服务，则可以创建别名资源记录集，与 CNAME 记录相比，该记录集具备 2 个优势。您可以在顶部节点 (example.com) 为域名创建别名资源记录集。此外，当您使用别名资源记录集时，就不用付款进行 Route 53 查询了。

> **Note**
>
> 如果您启用 IPv6，您必须创建两个别名资源记录集：一个用于路由 IPv4 流量 (A 记录)，一个用于路由 IPv6 流量 (AAAA 记录)。有关更多信息，请参阅 启用 IPv6主题中的 分配设置参考。

有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的使用域名将流量路由到 Amazon CloudFront Web 分配。

添加备用域名

以下任务列表描述了如何使用 CloudFront 控制台将备用域名添加到分配中，以便您可以在链接中使用自己的域名而非 CloudFront 域名。有关使用 CloudFront API 更新您的分配的信息，请参阅配置分配。

> **Note**
>
> 如果您希望查看器使用 HTTPS 和您的备用域名，请参阅 使用备用域名和 HTTPS。

开始之前：确保执行了以下操作，然后再更新分配以添加备用域名：

- 向 Route 53 或其他域注册商注册该域名。
- 从包含该域名的授权证书颁发机构 (CA) 获取 SSL/TLS 证书。将证书添加至您的分配中以验证您是否有权使用该域。有关更多信息，请参阅 使用备用域名的要求。

添加备用域名

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 为要更新的分配选择 ID。
3. 在 General 选项卡上，选择 Edit。
4. 更新以下值：

   备用域名 (CNAME)

   添加备用域名。用逗号隔开多个域名，或在新行中键入每个域名。

   SSL 证书

   选择以下设置：

   - 使用 HTTPS – 选择自定义 SSL 证书，然后从列表中选择证书。该列表包含 AWS Certificate Manager (ACM) 预置的证书、您从另一个证书颁发机构购买并上传到 ACM 的证书以及您从另一个证书颁发机构购买并上传到 IAM 证书存储的证书。

如果您已将证书上传到 IAM 证书存储，但该证书未显示在列表中，请检查过程 导入 SSL/TLS 证书，确认您已正确上传证书。

如果您选择此设置，建议您只在对象 URL 中使用一个备用域名 (https://www.example.com/logo.jpg)。如果您使用您的 CloudFront 分配域名 (https://d111111abcdef8.cloudfront.net.cloudfront.net/logo.jpg)，则查看器的工作方式可能如下所示，具体取决于您为支持的客户端选择的值：

- 所有客户端：如果查看器不支持 SNI，它将显示警告，因为 CloudFront 域名与 TLS/SSL 证书中的域名不匹配。
- 仅支持服务器名称指示 (SNI) 的客户端：CloudFront 将删除与查看器的连接而不返回对象。

支持的客户

选择一个选项：

- 所有客户端：CloudFront 使用专用 IP 地址提供您的 HTTPS 内容。如果您选择该选项，则当您将 SSL/TLS 证书与已启用的分配关联时，将产生额外的费用。有关更多信息，请参阅 Amazon CloudFront 定价。
- 仅限支持服务器名称指示 (SNI) 的客户端：不支持 SNI 的旧版浏览器或其他客户端必须使用其他方法访问您的内容。

有关更多信息，请参阅 选择 CloudFront 如何处理 HTTPS 请求。

5. 选择是，编辑。
6. 在分配的 General 选项卡上，确认 Distribution Status 已更改为 Deployed。如果您试图在部署对分配的更新之前使用备用域名，您在以下步骤创建的链接可能无法正常工作。
7. 配置备用域（例如 www.example.com）的 DNS 服务，以将流量路由到您的分配的 CloudFront 域名（如 d111111abcdef8.cloudfront.net）。您使用的方法取决于您是将 Route 53 还是另一个供应商作为域的 DNS 服务提供商。

> **Note**
>
> 如果您的 DNS 记录已指向的分配并不是您要更新的分配，则您只能在更新 DNS 后才能向分配中添加备用域名。有关更多信息，请参阅 备用域名的使用限制。

Route 53

创建别名资源记录。有了别名资源记录，您就不用付款进行 Route 53 查询了。此外，您可以为根域名 (example.com) 创建别名资源记录，而 DNS 不允许 CNAME。有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的使用域名将流量路由到 Amazon CloudFront Web 分配。

其他 DNS 服务提供商

使用 DNS 服务提供商提供的方法为域添加 CNAME 记录。这一新的 CNAME 记录将来自备用域名（例如，www.example.com）的 DNS 查询重新导向到分配的 CloudFront 域名（例如，d111111abcdef8.cloudfront.net）。有关更多信息，请参阅 DNS 服务提供商提供的文档。

> **Important**
>
> 如果您的域名已有现有的 CNAME 记录，请更新此记录或将其替换为指向分配的 CloudFront 域名的新记录。

8. 使用 `dig` 或类似的 DNS 工具，确认您在上一步创建的 DNS 配置指向分配的域名。

以下示例显示了 www.example.com 域上的 `dig` 请求以及响应的相关部分。

```
PROMPT> dig www.example.com

; <<> DiG 9.3.3rc2 <<> www.example.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0

;; QUESTION SECTION:
;www.example.com.      IN    A

;; ANSWER SECTION:
www.example.com. 10800 IN CNAME d111111abcdef8.cloudfront.net.
...
```

回答部分显示了将针对 www.example.com 的查询路由到 CloudFront 分配域名 d111111abcdef8.cloudfront.net 的 CNAME 记录。如果 CNAME 右侧的名称是 CloudFront 分配的

域名，则正确配置了 CNAME 记录。如果是其他任何值（如 Amazon S3 存储桶的域名），则未正确配置 CNAME 记录。在这种情况下，请回到步骤 7，纠正 CNAME 记录以指向分配的域名。

9. 通过访问具有您的域名而不是分配的 CloudFront 域名的 URL 来测试备用域名。
10. 在您的应用程序中，更改对象的 URL 以使用您的备用域名而不是 CloudFront 分配的域名。

将备用域名移动到其他分配

当您尝试为分配添加备用域名但备用域名已在其他分配中使用时，您会收到 `CNAMEAlreadyExists` 错误（您提供的一个或多个 CNAME 已与其他资源相关联）。例如，当您尝试将 www.example.com 添加到分配，但 www.example.com 已与其他分配相关联时，您会收到此错误。

在这种情况下，您需要将现有的备用域名从一个分配（源分配）移动到另一个分配（目标分配）。以下步骤概述了该过程。如需更多信息，请单击概览中每个步骤的链接。

移动备用域名

1. 设置目标分配。此分配必须有 SSL/TLS 证书，且该证书包含您要移动的备用域名。有关更多信息，请参阅设置目标分配。
2. 查找源分配。您可以使用 AWS Command Line Interface (AWS CLI) 查找与备用域名关联的分配。有关更多信息，请参阅查找源分配。
3. 移动备用域名。执行此操作的方式取决于源分配和目标分配是否在同一 AWS 账户中。有关更多信息，请参阅the section called “移动备用域名”。

设置目标分配

移动备用域名之前，您必须设置目标分配，也就是您要将备用域名移动到的分配。

设置目标分配

1. 获取包含您要移动的备用域名的 SSL/TLS 证书。如果没有，您可以从 AWS Certificate Manager (ACM) 请求一个，或从其他证书颁发机构 (CA) 获取一个并将其导入 ACM。请确保您在美国东部（弗吉尼亚州北部）(`us-east-1`) 区域请求或导入证书。
2. 如果您尚未创建目标分配，请立即创建一个。将您的证书（来自上一步）与分配相关联，这是创建目标分配的一个环节。有关更多信息，请参阅创建分配。

   如果您已经有目标分配，请将您的证书（来自上一步）与目标分配相关联。有关更多信息，请参阅更新分配。

3. 创建 DNS TXT 记录，将备用域名与目标分配的分配域名关联起来。创建在备用域名前面添加下划线 (_) 的 TXT 记录。以下为 DNS 中的 TXT 记录示例：

   ```
   _www.example.com TXT d111111abcdef8.cloudfront.net
   ```

   CloudFront 使用此 TXT 记录来验证您对备用域名的所有权。

查找源分配

在将备用域名从一个分配移动到另一个分配之前，您应该找到源分配（备用域名当前正在使用的分配）。获取源和目标分配的AWS账户 ID 后，您即可确定如何移动备用域名。

查找备用域名的源分配

1. 在 AWS Command Line Interface (AWS CLI)中使用 CloudFront list-conflicting-aliases 命令，如下例所示。将 *www.example.com* 替换为备用域名，将 *EDFDVBD6EXAMPLE* 替换为 您之前设置的目标分配的 ID。使用与目标分配在同一AWS账户中的凭据运行此命令。使用此命令的前提是，您必须拥有目标分配的 `cloudfront:GetDistribution` 和 `cloudfront:ListConflictingAlias` 权限。

   ```
   aws cloudfront list-conflicting-aliases --alias www.example.com --distribution-
   id EDFDVBD6EXAMPLE
   ```

   该命令的输出显示了与所提供的域名冲突或重叠的所有备用域名列表。例如：

   - 如果您向命令提供 www.example.com，则命令的输出包括 www.example.com 和重叠的通配符备用域名 (*.example.com)（如果存在）。
   - 如果您向命令提供 *.example.com，则命令的输出包括 *.example.com 和该通配符涵盖的所有备用域名（例如，www.example.com、test.example.com、dev.example.com 等）。

   对于命令输出中的每个备用域名，您可以查找与其关联的分配的 ID，以及拥有此分配的AWS账户 ID。分配和账户 ID 信息是部分隐藏的，这使您可以识别您拥有的分配和账户，并有助于保护您没有所有权的分配和账户的信息。

2. 在命令的输出中，找到您要移动的备用域名的分配，并记下源分配的AWS账户 ID。将源分配的账户 ID 与您创建目标分配的账户 ID 进行比较，并确定这两个分配是否在同一AWS账户中。这有助于您确定如何移动备用域名。

要移动备用域名，请参阅以下主题。

移动备用域名

请根据您的具体情况，从以下方式中选择移动备用域名的方式：

如果源分配和目标分配在同一AWS账户中

使用 AWS CLI 中的 associate-alias 命令移动备用域名。此方法适用于所有同账户内移动，包括当备用域名是顶级域（也称为根域，如 example.com）时。有关更多信息，请参阅the section called “使用 associate-alias 移动备用域名”。

如果源分配和目标分配在不同AWS账户中

如果您有权访问源分配，备用域名不是顶级域（也称为根域，如 example.com），并且您尚未使用与该备用域名重叠的通配符，请使用通配符移动备用域名。有关更多信息，请参阅 the section called “使用通配符移动备用域名”。

如果您无法访问源分配的AWS账户，则可以尝试使用 AWS CLI 中的 associate-alias 命令移动备用域名。如果源分配已禁用，则您可以移动备用域名。有关更多信息，请参阅 the section called “使用 associate-alias 移动备用域名”。如果 associate-alias 命令不起作用，请联系 AWS Support。有关更多信息，请参阅 the section called “联系 AWS Support 以移动备用域名”。

使用 **associate-alias** 移动备用域名

如果源分配与目标分配在同一 AWS 账户中，或者如果源分配在不同的账户中但已被禁用，您可以使用 AWS CLI 中的 CloudFront associate-alias命令移动备用域名。

使用关联别名移动备用域名

1. 使用 AWS CLI 运行 CloudFront associate-alias 命令，如下例所示。将 *www.example.com* 替换为备用域名，将 *EDFDVBD6EXAMPLE* 替换为目标分配 ID。使用与目标分配在同一AWS账户中的凭据运行此命令。使用此命令时，请注意以下限制：

   - 您必须拥有目标分配的 `cloudfront:AssociateAlias` 和 `cloudfront:UpdateDistribution` 权限。
   - 如果源分配和目标分配在同一AWS账户中，您必须对源分配具有 `cloudfront:UpdateDistribution` 权限。
   - 如果源分配和目标分配在不同AWS账户中，则必须禁用源分配。

- 必须按照the section called “设置目标分配”中所述设置目标分配。

```
aws cloudfront associate-alias --alias www.example.com --target-distribution-
id EDFDVBD6EXAMPLE
```

此命令通过从源分配中删除备用域名并将其添加到目标分配，来更新这两个分配。

2. 目标分配完全部署后，更新您的 DNS 配置以将备用域名的 DNS 记录指向目标分配的分配域名。

使用通配符移动备用域名

如果源分配与目标分配在不同AWS账户中，并且源分配已启用，您可以使用通配符移动备用域名。

> **Note**
>
> 您不能使用通配符移动顶级域（例如 example.com）。如果源分配和目标分配在不同AWS账户中，要移动顶点域，请联系 AWS Support。有关更多信息，请参阅the section called “联系 AWS Support 以移动备用域名”。

使用通配符移动备用域名

> **Note**
>
> 此过程涉及对分配的多次更新。在继续下一步之前，请等待每个分配完全部署完最新的更改。

1. 更新目标分配以添加一个通配符备用域名，该域名涵盖您要移动的备用域名。例如，如果您要移动的备用域名是 www.example.com，请将备用域名 *.example.com 添加到目标分配中。为此，目标分配上的 SSL/TLS 证书必须包含通配符域名。有关更多信息，请参阅the section called “更新分配”。
2. 更新备用域名的 DNS 设置以指向目标分配的域名。例如，如果您要移动的备用域名是 www.example.com，请更新 www.example.com 的 DNS 记录以将流量路由到目标分配的域名（例如 d111111abcdef8.cloudfront.net）。

> **Note**
>
> 即使在您更新 DNS 设置之后，备用域名仍由源分配提供服务，因为这是当前配置备用域名的位置。

3. 更新源分配以删除备用域名。有关更多信息，请参阅更新分配。
4. 更新目标分配以添加备用域名。有关更多信息，请参阅更新分配。
5. 使用 dig（或类似的 DNS 查询工具）验证备用域名的 DNS 记录是否解析为目标分配的域名。
6. （可选）更新目标分配以删除通配符备用域名。

联系 AWS Support 以移动备用域名

如果源分配和目标分配在不同 AWS 账户中，并且您无权访问源分配的 AWS 账户或无法禁用源分配，您可以联系 AWS Support 来移动备用域名。

联系 AWS Support 以移动备用域名

1. 设置目标分配，包括指向目标分配的 DNS TXT 记录。有关更多信息，请参阅设置目标分配。
2. 联系 AWS Support 以请求他们验证您是否拥有该域，然后为您将该域移动到新的 CloudFront 分配。
3. 目标分配完全部署后，更新您的 DNS 配置以将备用域名的 DNS 记录指向目标分配的分配域名。

删除备用域名

如果您要停止将域或子域的流量路由到 CloudFront 分配，请按本节中的步骤更新 DNS 配置和 CloudFront 分配。

请务必从该分配中删除备用域名并更新您的 DNS 配置。当您希望将域名与其他 CloudFront 分配关联时，这有助于防止以后出现问题。如果某个备用域名已经与一个分配关联，则不能将该域名设置为与其他分配关联。

> **Note**
>
> 如果您要从此分配中删除备用域名以便能将它添加到其他分配，请按照将备用域名移动到其他分配中的步骤操作。如果您改为执行此处的步骤（用于删除域），然后将域添加到另一个分配，则在一段时间内，域不会链接到新分配，因为 CloudFront 会传播到对边缘站点的更新。

从分配中删除备用域名

1. 要开始操作，请将您的域的 Internet 流量路由到不是您的 CloudFront 分配的其他资源（如 Elastic Load Balancing 负载均衡器）。或者，您也可以删除将流量路由到 CloudFront 的 DNS 记录。

   根据您的域的 DNS 服务执行以下操作之一：

   - 如果您使用的是 Route 53，请更新或删除别名记录或 CNAME 记录。有关更多信息，请参阅编辑记录或删除记录。
   - 如果您使用的是其他 DNS 服务提供商，请使用 DNS 服务提供商提供的方法来更新或删除将流量引向 CloudFront 的 CNAME 记录。有关更多信息，请参阅 DNS 服务提供商提供的文档。
2. 在更新您的域的 DNS 记录后，请等到更改已传播且 DNS 解析程序正在将流量路由到新资源。您可以通过创建一些在 URL 中使用您的域的测试链接来检查确认该操作完成的时间。
3. 登录到AWS Management Console并打开 CloudFront 控制台 (https://console.aws.amazon.com/cloudfront/v4/home)，然后更新您的 CloudFront 分配并通过执行以下操作来删除域名：

   a. 为要更新的分配选择 ID。
   b. 在 General 选项卡上，选择 Edit。
   c. 在备用域名(CNAME) 中，删除您不想再用于分配的备用域名（或域名）。
   d. 选择是，编辑。

在备用域名中使用通配符

在添加备用域名时，可以在域名开头使用 * 通配符，而不是逐个添加子域。例如，如果备用域名为 *.example.com，您可以在 URL 中使用以 example.com 结尾的任何域名，例如 www.example.com、product-name.example.com、marketing.product-name.example.com 等。无论域名如何，对象的路径都相同，例如：

- www.example.com/images/image.jpg
- product-name.example.com/images/image.jpg
- marketing.product-name.example.com/images/image.jpg

对于包含通配符的备用域名，请遵循以下要求：

- 备用域名必须以星号和点 (*.) 开头。
- 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。
- 所有备用域名，包括使用通配符的备用域名，必须涵盖在证书的使用者备用名称 (SAN) 中。

通配符备用域名（例如 *.example.com）可以包含其他正在使用的备用域名，例如 example.com。

将 WebSocket 与 CloudFront 分配结合使用

Amazon CloudFront 支持使用 WebSocket，后者是一种基于 TCP 的协议，它在客户端和服务器之间需要长期双向连接时很有用。对于实时应用程序，持久连接通常是必需的。可使用 Websocket 的场景包括社交聊天平台、在线协作工作区、多玩家游戏和提供实时数据馈送（如金融贸易平台）的服务。对于全双工通信，通过 WebSocket 连接的数据可双向流动。

WebSocket 功能会自动启用，以便与任何分配结合使用。要使用 WebSocket，请在附加到您的分配的缓存行为中配置以下内容之一：

- 将所有查看器请求标头转发到源。（您可以使用 AllViewer 托管源请求策略。）
- 具体而言，就是在您的源请求策略中转发 `Sec-WebSocket-Key` 和 `Sec-WebSocket-Version` 请求标头。

WebSocket 协议的工作原理

WebSocket 协议是一种独立的、基于 TCP 的协议，可让您避免 HTTP 的一些开销和潜在的延迟增加。

为了建立 WebSocket 连接，客户端将发送使用 HTTP 升级语义更改协议的常规 HTTP 请求。随后，服务器可以完成握手。WebSocket 连接将保持打开状态，并且客户端或服务器可以相互发送数据帧，而无需每次都建立新连接。

默认情况下，WebSocket 协议使用端口 80（对于常规 WebSocket 连接）和端口 443（对于通过 TLS/SSL 的 WebSocket 连接）。您为 CloudFront 查看器协议策略 和 协议（仅自定义源） 选择的选项适用于 WebSocket 连接以及 HTTP 流量。

WebSocket 要求

WebSocket 请求必须遵守 RFC 6455（采用以下标准格式）。

示例客户端请求：

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: https://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

示例服务器响应：

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

如果客户端或服务器或网络中断已断开 WebSocket 连接，则客户端应用程序应重新发起与服务器的连接。

推荐的 WebSocket 标头

为了避免在使用 WebSocket 时出现与压缩相关的意外问题，建议您在源请求策略中包含以下标头：

- `Sec-WebSocket-Key`
- `Sec-WebSocket-Version`
- `Sec-WebSocket-Protocol`
- `Sec-WebSocket-Accept`
- `Sec-WebSocket-Extensions`

缓存和可用性

您可以使用 CloudFront 来减少原始服务器必须直接响应的请求数量。借助 CloudFront 缓存，更多对象可以从更靠近用户的 CloudFront 边缘站点提供。这既减少了源服务器上的负载，也减少了延迟。

CloudFront 可以通过边缘缓存提供服务的请求越多，CloudFront 为获取对象的最新版本或唯一版本而需转发到源的查看器请求就越少。要优化 CloudFront 以尽可能少地向您的源发出请求，请考虑使用 CloudFront Origin Shield。有关更多信息，请参阅 使用 Amazon CloudFront Origin Shield。

直接从 CloudFront 缓存提供服务的请求与所有请求相比的比例称为缓存命中率。您可在 CloudFront 控制台中查看命中、未命中和出错的查看器请求百分比。有关更多信息，请参阅 查看 CloudFront 缓存统计报告。

很多因素都会影响缓存命中率。您可以按照增加直接从 CloudFront 缓存提供服务的请求的比例（缓存命中率）中的指导操作来调整您的 CloudFront 分配配置以提高缓存命中率。

要了解添加和删除您希望 CloudFront 提供的内容的信息，请参阅添加、删除或替换 CloudFront 分配的内容。

主题

增加直接从 CloudFront 缓存提供服务的请求的比例（缓存命中率）

您可以通过增加直接从 CloudFront 缓存提供服务的查看器请求的比例（而不是转至源服务器以获得内容）来提高性能。这称为提高缓存命中率。

以下部分说明了如何提高缓存命中率。

主题

指定 CloudFront 缓存对象的时间长度

要提升缓存命中率，您可以配置源来将 Cache-Control max-age 指令添加到您的对象，并为 `max-age` 指定在实践中可行的最长值。缓存持续时间越短，CloudFront 将请求发送到源（用以确定对象是否已更改，并获取最新版本）的次数就越多。您可以用 `stale-while-revalidate` 和 `stale-if-error` 指令来补充 `max-age`，以进一步提高某些条件下的缓存命中率。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

使用源护盾

Origin Shield 可以帮助提高 CloudFront 分配的缓存命中率，因为它在源之前提供了额外的缓存层。当您使用 Origin Shield 时，从所有 CloudFront 缓存层到您的源的所有请求都来自一个位置。CloudFront 可以使用来自 Origin Shield 的单个源请求检索每个对象，CloudFront 缓存的所有其他层（边缘站点和区域边缘缓存）可以从 Origin Shield 中检索对象。

有关更多信息，请参阅 使用 Amazon CloudFront Origin Shield。

根据查询字符串参数进行缓存

如果您将 CloudFront 配置为基于查询字符串参数进行缓存，则可以通过以下方法来改进缓存：

- 将 CloudFront 配置为仅转发您的源将返回唯一对象的那些查询字符串参数。
- 为相同参数的所有实例使用相同的大小写。例如，如果一个请求包含 `parameter1=A`，另一个请求包含 `parameter1=a`，则 CloudFront 在一个请求包含 `parameter1=A` 而一个请求包含 `parameter1=a` 时，会将两个请求分别转发到源。然后，CloudFront 分别缓存源返回的对应对象，即使这些对象完全相同。如果您仅使用 `A` 或 `a`，CloudFront 会将较少的请求转发到源。
- 按相同的顺序列出参数。与大小写不同的情况相似，如果对象的一个请求包含查询字符串 `parameter1=a&parameter2=b`，而相同对象的另一个请求包含

parameter2=b&parameter1=a，CloudFront 会将两个请求转发到源，并分别缓存对应的对象，即使它们完全相同。如果您始终为参数使用相同的顺序，CloudFront 会将较少的请求转发到源。

有关更多信息，请参阅 根据查询字符串参数缓存内容。如果您希望查看 CloudFront 转发到源的查询字符串，请查看 CloudFront 日志文件的 cs-uri-query 列中的值。有关更多信息，请参阅 配置和使用标准日志（访问日志）。

根据 Cookie 值进行缓存

如果您将 CloudFront 配置为基于 Cookie 值进行缓存，则可以通过以下方法来改进缓存：

- 将 CloudFront 配置为仅转发指定的 Cookie 而不是转发所有 Cookie。对于配置为由 CloudFront 转发到源的 Cookie，CloudFront 将转发每个 Cookie 名称和值组合。然后，它单独缓存由源返回的对象，即使这些对象全都相同。

  例如，假设查看器在每个请求中包含两个 Cookie，每个 Cookie 有三个可能的值，并且 Cookie 值的所有组合均可能。CloudFront 最多会为每个对象将六个不同的请求转发到您的源。如果您的源仅基于其中一个 Cookie 返回不同的对象版本，则 CloudFront 会不必要地将更多请求转发到您的源，并且不必要地缓存对象的多个相同版本。
- 为静态和动态内容创建单独的缓存行为，将 CloudFront 配置成仅为动态内容将 Cookie 转发到源。

  举例而言，假设您的分配只有一个缓存行为，而您希望将分配同时用于动态内容（例如 .js 文件）和很少更改的 .css 文件。CloudFront 根据 Cookie 值缓存 .css 文件的单独版本，这样每个 CloudFront 边缘站点会为每个新 Cookie 值或 Cookie 值的组合将请求转发到您的源。

  如果您创建一个缓存行为，其路径模式为 *.css 并且 CloudFront 不根据 Cookie 值进行缓存，则 CloudFront 仅在以下情况下将 .css 文件的请求转发到源：边缘站点收到给定 .css 文件的第一个请求以及 .css 文件过期后收到的第一个请求。
- 如果可能，请在 Cookie 值对于每个用户唯一（例如用户 ID）时为动态内容创建单独的缓存行为，并为基于少量唯一值变化的动态内容创建单独的缓存行为。

有关更多信息，请参阅 根据 Cookie 缓存内容。如果要查看 CloudFront 转发到源的 Cookie，请查看 CloudFront 日志文件的 cs(Cookie) 列中的值。有关更多信息，请参阅 配置和使用标准日志（访问日志）。

根据请求标头进行缓存

如果您将 CloudFront 配置为基于请求标头进行缓存，则可以通过以下方法来改进缓存：

- 将 CloudFront 配置为仅基于指定标头进行转发和缓存，而不是基于所有标头转发和缓存。对于您指定的标头，CloudFront 将转发每个标头名称和值组合。然后，它单独缓存由源返回的对象，即使这些对象全都相同。

  > **Note**
  >
  > CloudFront 始终将以下主题中指定的标头转发给您的源：
  >
  > - CloudFront 如何处理请求并将请求转发到您的 Amazon S3 原始服务器 > CloudFront 删除或更新的 HTTP 请求标头
  > - CloudFront 如何处理请求及如何将请求转发给您的自定义原始服务器 > HTTP 请求标头和 CloudFront 行为（自定义源和 Amazon S3 源）

  当您将 CloudFront 配置为基于请求标头进行缓存时，无需更改 CloudFront 转发的标头，仅更改 CloudFront 是否基于标头值缓存对象。

- 尝试避免基于具有大量唯一值的请求标头进行缓存。

  例如，如果您希望基于用户设备提供不同大小的图像，不要将 CloudFront 配置为基于 `User-Agent` 标头进行缓存，这会导致大量可能的值。而应该将 CloudFront 配置为基于 CloudFront device-type 标头 `CloudFront-Is-Desktop-Viewer`、`CloudFront-Is-Mobile-Viewer`、`CloudFront-Is-SmartTV-Viewer` 和 `CloudFront-Is-Tablet-Viewer` 进行缓存。此外，如果您为平板电脑和桌面返回相同版本的图像，则仅转发 `CloudFront-Is-Tablet-Viewer` 标头，而不是 `CloudFront-Is-Desktop-Viewer` 标头。

有关更多信息，请参阅 根据请求标头缓存内容。

不需要压缩时删除 `Accept-Encoding` 标头

如果未启用压缩 – 由于源不支持它，CloudFront 不支持它，或者内容不可压缩 – 您可以通过将分配中的缓存行为与设置 Custom Origin Header 的源相关联来增加缓存命中率，如下所示：

- 标头名称：`Accept-Encoding`
- 标头值：（保留为空）

在使用此配置时，CloudFront 会从缓存键中删除 `Accept-Encoding` 标头，并且不在源请求中包含标头。此配置应用于 CloudFront 从该源通过分发所提供的所有内容。

通过 HTTP 提供媒体内容

有关优化点播视频 (VOD) 和流式传输视频内容的信息，请参阅使用 CloudFront 的点播视频和实时流视频。

使用 Amazon CloudFront Origin Shield

CloudFront Origin Shield 是 CloudFront 缓存基础设施中的一个附加层，有助于最大限度地减少源的负载、提高其可用性并降低其运营成本。借助 CloudFront Origin Shield，您可以获得以下优势：

更好的缓存命中率

Origin Shield 可以帮助提高 CloudFront 分配的缓存命中率，因为它在源之前提供了额外的缓存层。当您使用 Origin Shield 时，从 CloudFront 缓存层到源的所有请求都会通过 Origin Shield，从而增加缓存命中的可能性。CloudFront 可以使用从 Origin Shield 到您的源的单个源请求检索每个对象，CloudFront 缓存的所有其他层（边缘站点和区域边缘缓存）可以从 Origin Shield 中检索对象。

减少源负荷

Origin Shield 可以进一步减少针对同一对象发送到源的同时请求的数量。对不在 Origin Shield 缓存中的内容的请求将与对同一对象的其他请求合并，从而导致发往源的请求只有一个。在源中处理较少的请求可以在高峰负载或意外流量峰值期间保持源的可用性，并且可以降低即时打包、图像转换和数据传出 (DTO) 等方面的成本。

更好的网络性能

当您在与源之间的延迟最短的 AWS 区域中启用 Origin Shield 时，可以获得更好的网络性能。对于 AWS 区域中的源，CloudFront 网络流量始终保留在高吞吐量 CloudFront 网络上，直至您的源。对于 AWS 外部的源，CloudFront 网络流量一直保留在 CloudFront 网络上，直至到达 Origin Shield（它与源之间具有低延迟连接）。

使用 Origin Shield 需要支付额外费用。有关更多信息，请参阅 CloudFront 定价。

主题

- Origin Shield 高可用性
- Origin Shield 如何与其他 CloudFront 功能进行交互

Origin Shield 的使用案例

CloudFront Origin Shield 适用于许多使用案例，包括以下情况：

- 分布在不同地理区域的查看器
- 为实时流式处理或动态图像处理提供即时打包的源
- 具有容量或带宽限制的内部源
- 使用多个内容分发网络 (CDN) 的工作负载

Origin Shield 可能不太适合其他情况，例如通过代理到达源的动态内容、可缓存性低的内容或不经常请求的内容。

以下各节介绍 Origin Shield 在以下使用案例中的优势。

使用案例

- 位于不同地理区域的查看器
- 多个 CDN

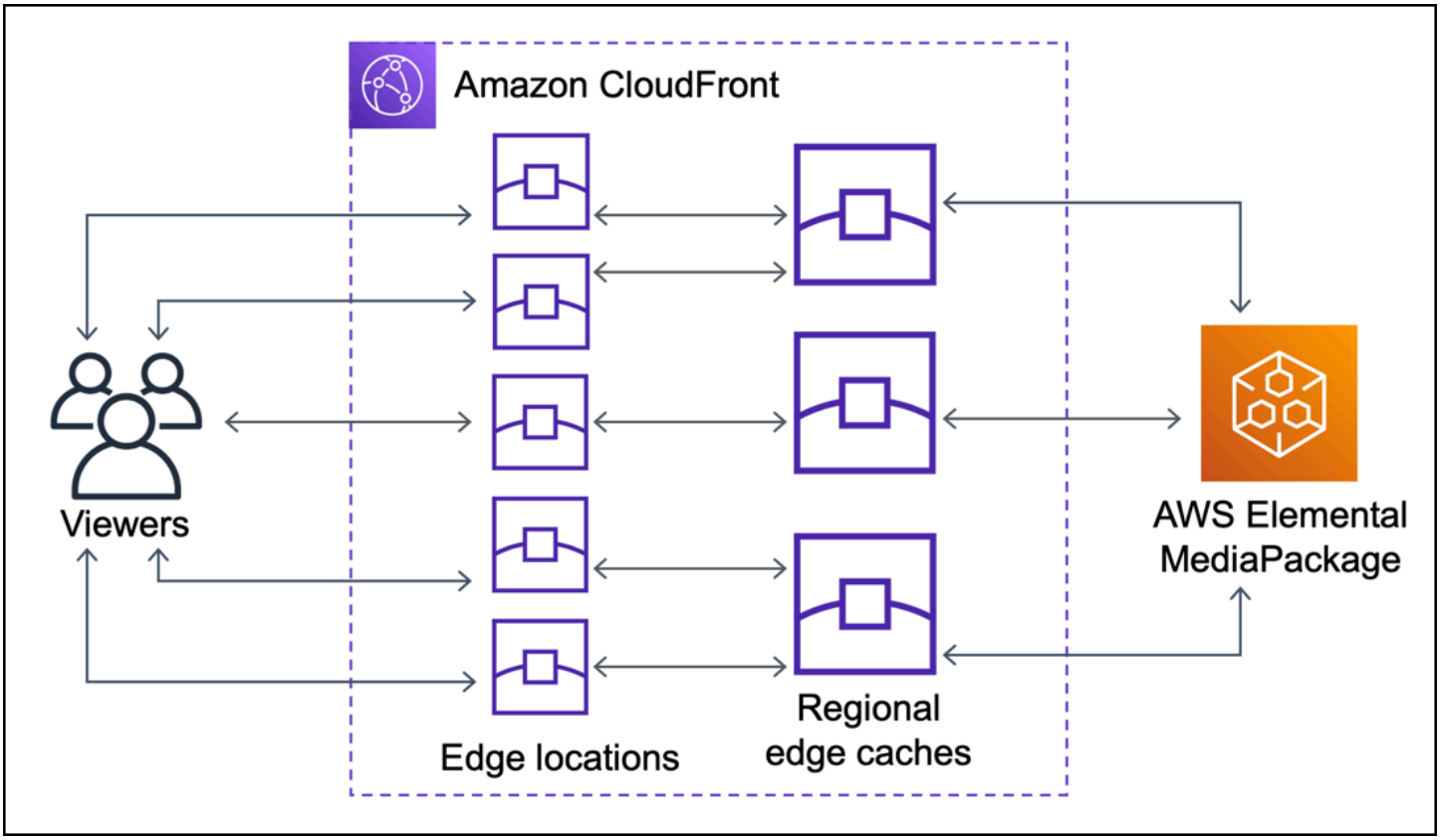
位于不同地理区域的查看器

通过 Amazon CloudFront，您本质上可以减少源上的负载，因为 CloudFront 可以从缓存提供服务的请求不会转到源。除了 CloudFront 的全球边缘站点网络外，区域边缘缓存还作为中间层缓存层，为附近地理区域中的查看器提供缓存命中并合并源请求。查看器请求首先路由到附近的 CloudFront 边缘站点，如果对象未在该站点缓存，则请求将发送到区域边缘缓存。

当查看器位于不同的地理区域时，请求可以通过不同的区域边缘缓存进行路由，其中每个缓存都可以向您的源发送对相同内容的请求。但使用 Origin Shield，您可以在区域边缘缓存和源之间获得额外的缓存层。来自所有区域边缘缓存的所有请求都经过 Origin Shield，从而进一步减少源上的负载。下面的示意图对此进行说明。在下图中，源为 AWS Elemental MediaPackage。

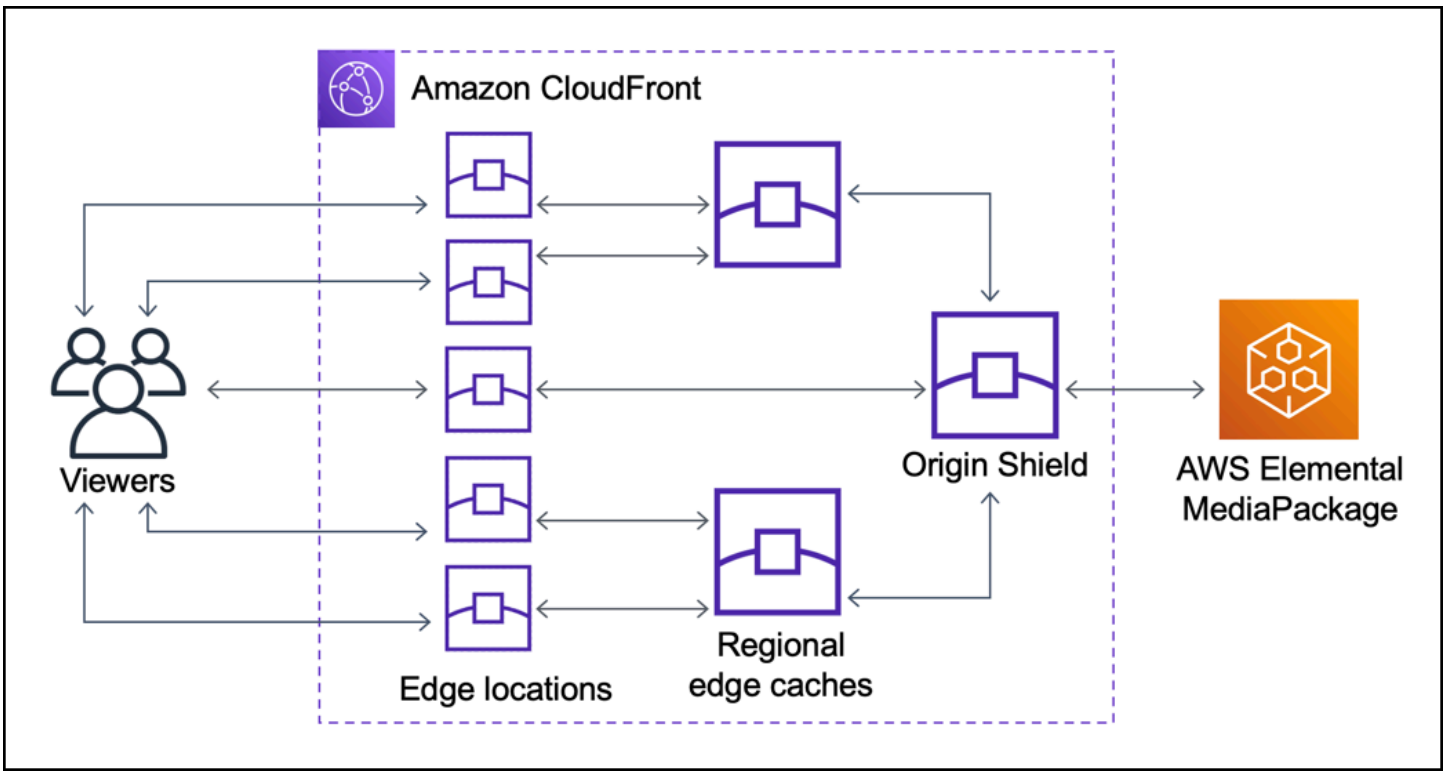
不使用 Origin Shield

如果没有 Origin Shield，源可能会收到针对相同内容的重复请求，如下图所示。



使用 Origin Shield

使用 Origin Shield 有助于减少源上的载荷，如下图所示。



多个 CDN

要提供实时视频活动或热门点播内容，您可以使用多个内容分发网络 (CDN)。使用多个 CDN 可以提供某些优势，但这也意味着源可能会收到许多对同一内容的重复请求，每个请求来自不同的 CDN 或同一 CDN 内的不同位置。这些冗余请求可能会对源的可用性产生不利影响，或导致进程（如即时打包或数据传出 (DTO) 到 Internet）的额外运营成本。

当您将 Origin Shield 与将 CloudFront 分配用作其他 CDN 的源相结合时，可以获得以下好处：

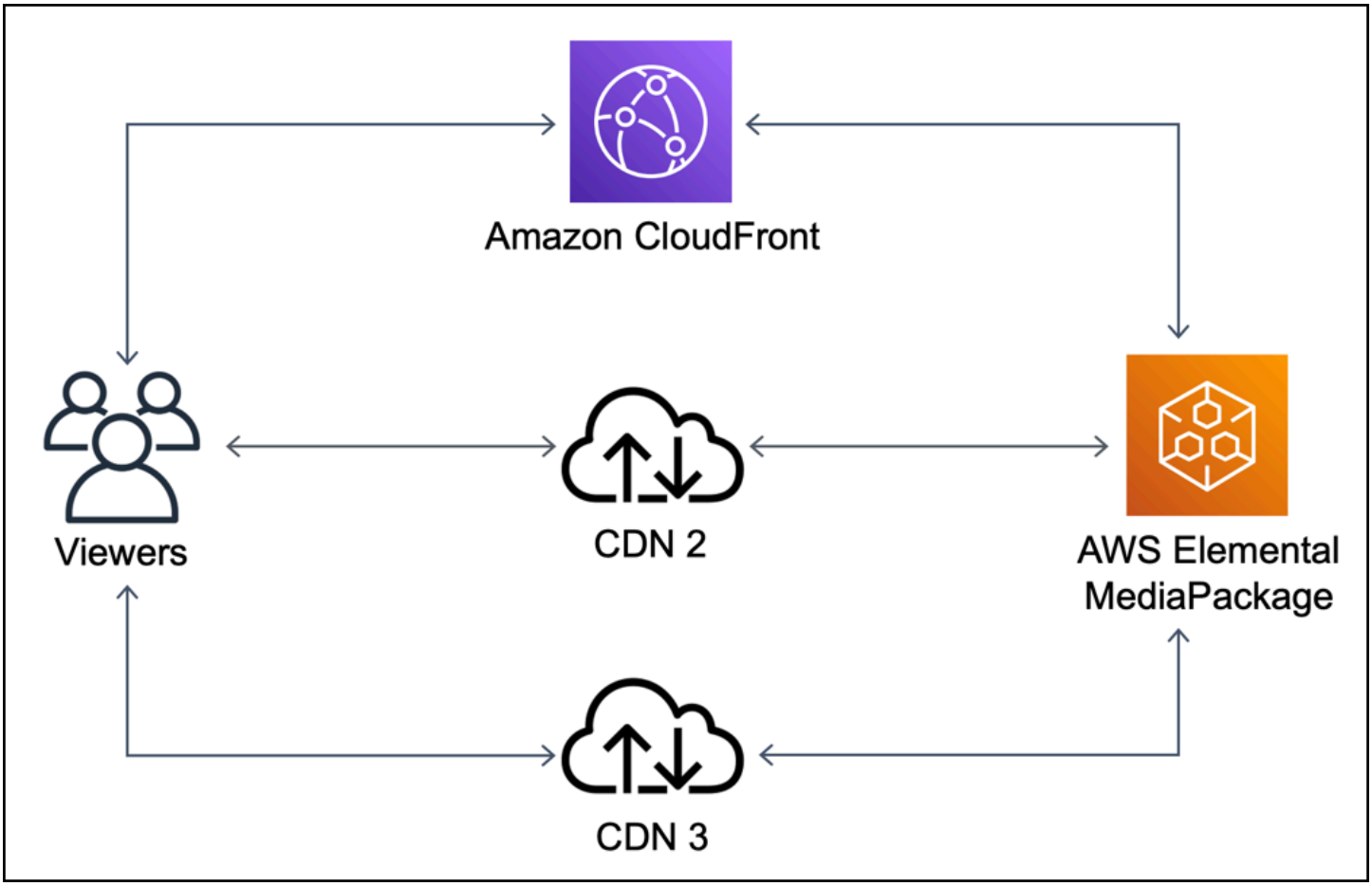
- 减少源收到的冗余请求数量，这有助于减少使用多个 CDN 的负面影响。
- 跨 CDN 的通用缓存密钥，以及面向源的功能的集中管理。
- 提高了网络性能。来自其他 CDN 的网络流量在附近的 CloudFront 边缘站点终止，这可能会提供来自本地缓存的命中。如果请求的对象不在边缘站点缓存中，则对源的请求将一直保留在 CloudFront 网络上直至 Origin Shield，从而为源提供高吞吐量和低延迟。如果请求的对象位于 Origin Shield 的缓存中，则完全避免对源的请求。

> **Important**
>
> 如果您有兴趣在多 CDN 架构中使用 Origin Shield，并且享有折扣价格，请联系我们或您的 AWS 销售代表以获取更多信息。可能收取额外费用。

下图显示了当您使用多个 CDN 提供流行的实时视频活动时，此配置如何帮助最大限度地减少源上的负载。在下图中，源为 AWS Elemental MediaPackage。

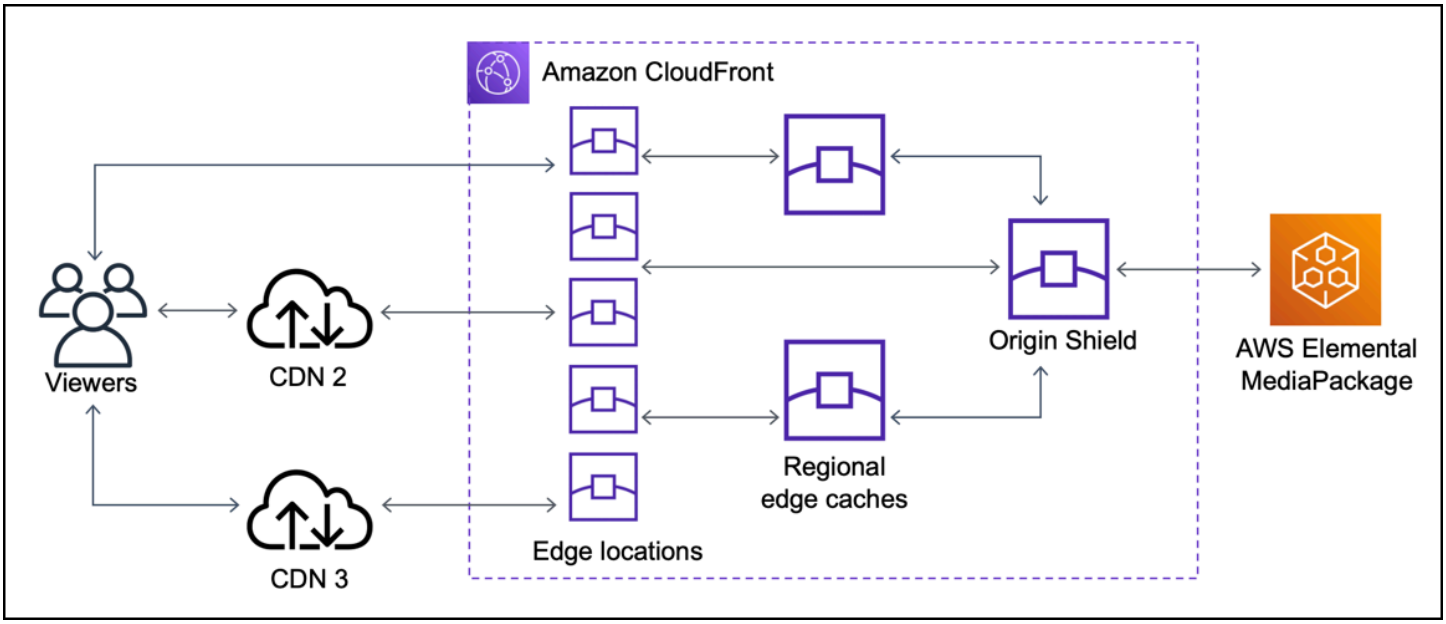
不使用 Origin Shield（多个 CDN）

如果不使用 Origin Shield，源可能会收到针对相同内容的重复请求，每个请求来自不同的 CDN，如下图所示。



使用 Origin Shield（多个 CDN）

使用 Origin Shield（将 CloudFront 作为其他 CDN 的源）有助于减少源上的载荷，如下图所示。



为 Origin Shield 选择 AWS 区域

在 CloudFront 具有区域性边缘缓存的 AWS 区域中，Amazon CloudFront 提供 Origin Shield。启用 Origin Shield 后，可以为 Origin Shield 选择 AWS 区域。您应该选择与源之间具有最低延迟的 AWS 区域。您可以将 Origin Shield 与位于 AWS 区域中的源和不在 AWS 中的源结合使用。

对于 AWS 区域中的源

如果您的源位于 AWS 区域，请首先确定您的源是否位于 CloudFront 提供 Origin Shield 的区域。CloudFront 在以下 AWS 区域提供 Origin Shield。

- 美国东部（俄亥俄州）– (us-east-2)
- 美国东部（弗吉尼亚州北部）– us-east-1
- 美国西部（俄勒冈州）– us-west-2
- 亚太地区（孟买）– (ap-south-1)
- 亚太地区（首尔）– (ap-northeast-2)
- 亚太地区（新加坡）– (ap-southeast-1)
- 亚太地区（悉尼）– ap-southeast-2
- 亚太地区（东京）– (ap-northeast-1)
- 欧洲地区（法兰克福）– eu-central-1
- 欧洲地区（爱尔兰）– eu-west-1
- 欧洲地区（伦敦）– eu-west-2

- 南美洲（圣保罗）– (sa-east-1)

如果您的源位于 CloudFront 提供 Origin Shield 的AWS区域

如果源位于 CloudFront 提供 Origin Shield 的 AWS 区域中（请参阅前面的列表），请在与源相同的区域中启用 Origin Shield。

如果您的源不在 CloudFront 提供 Origin Shield 的AWS区域

如果源不在 CloudFront 提供 Origin Shield 的 AWS 区域中，请参阅下表以确定要在哪个区域中启用 Origin Shield。

如果源位于...	启用 Origin Shield 的位置...
美国西部（加利福尼亚北部）– us-west-1	美国西部（俄勒冈州）– us-west-2
非洲（开普敦）– af-south-1	欧洲地区（爱尔兰）– eu-west-1
亚太地区（香港）– ap-east-1	亚太地区（新加坡）– (ap-southeast-1)
加拿大（中部）– ca-central-1	美国东部（弗吉尼亚州北部）– us-east-1
欧洲地区（米兰）– eu-south-1	欧洲地区（法兰克福）– eu-central-1
欧洲地区（巴黎）– eu-west-3	欧洲地区（伦敦）– eu-west-2
欧洲地区（斯德哥尔摩）– eu-north-1	欧洲地区（伦敦）– eu-west-2
中东（巴林）– me-south-1	亚太地区（孟买）– (ap-south-1)

对于之外的源AWS

您可以将 Origin Shield 与本地或不在 AWS 区域中的源结合使用。在这种情况下，请在与源之间的延迟最低的 AWS 区域中启用 Origin Shield。如果您不确定哪个 AWS 区域与源之间的延迟最低，您可以使用以下建议来帮助做出决定。

- 您可以参阅上表，根据源的地理位置，了解哪个 AWS 区域可能与源之间具有最低延迟的近似情况。
- 您可以在地理位置靠近源的几个不同 AWS 区域中启动 Amazon EC2 实例，并使用 ping 运行一些测试来测量这些区域与源之间的典型网络延迟。

启用 Origin Shield

您可以启用 Origin Shield 来提高缓存命中率，减少源上的负载，并帮助提高性能。要启用 Origin Shield，请更改 CloudFront 分配中的源设置。Origin Shield 是源的一个属性。对于 CloudFront 分配中的每个源，您可以在为该源提供最佳性能的任何 AWS 区域中单独启用 Origin Shield。

您可以使用 AWS CloudFormation 或 CloudFront API 启用 CloudFront 控制台中的 Origin Shield。

Console

为现有源启用 Origin Shield（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择具有要更新的源的分配。
3. 选择源和源组选项卡。
4. 选择要更新的源，然后选择编辑。
5. 对于启用 Origin Shield ，选择是。
6. 对于 Origin Shield 区域，选择要在其中启用 Origin Shield 的 AWS 区域。有关选择区域的帮助，请参阅为 Origin Shield 选择 AWS 区域。
7. 在页面底部选择是，编辑。

当您的分配状态为已部署时，Origin Shield 已准备就绪。这需要几分钟。

为新源启用 Origin Shield（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 要在现有分配中创建新源，请执行以下操作：

   1. 选择要在其中创建源的分配。
   2. 选择创建源，然后继续执行步骤 3。

   要在新分配中创建新源，请执行以下操作：

   1. 选择 Create Distribution。
   2. 在 Web 部分中，选择入门。在源设置部分，完成以下步骤，从步骤 3 开始。

3. 对于启用 Origin Shield ，选择是。
4. 对于 Origin Shield 区域，选择要在其中启用 Origin Shield 的 AWS 区域。有关选择区域的帮助，请参阅为 Origin Shield 选择 AWS 区域。

   如果要创建新的分配，请使用页面上的其他设置继续配置分配。有关更多信息，请参阅 分配设置参考。
5. 请确保通过选择创建（对于现有分配中的新源）或创建分配（对于新分配中的新源）来保存更改。

当您的分配状态为已部署时，Origin Shield 已准备就绪。这需要几分钟。

AWS CloudFormation

要使用 AWS CloudFormation 启用 Origin Shield，请使用 `OriginShield` 资源的 `Origin` 属性类型中的 `AWS::CloudFront::Distribution` 属性。您可以将 `OriginShield` 属性添加到现有 `Origin` 属性，也可以在创建新的 `Origin` 属性时将其包括在内。

以下示例以 YAML 格式显示在美国西部（俄勒冈州）区域 (`OriginShield`) 中启用 `us-west-2` 的语法。有关选择区域的帮助，请参阅the section called “为 Origin Shield 选择 AWS 区域”。此示例仅显示 `Origin` 属性类型，而不显示整个 `AWS::CloudFront::Distribution` 资源。

```
Origins:
- DomainName: 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com
  Id: Example-EMP-3ae97e9482b0d011
  OriginShield:
    Enabled: true
    OriginShieldRegion: us-west-2
  CustomOriginConfig:
    OriginProtocolPolicy: match-viewer
    OriginSSLProtocols: TLSv1
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》的资源和属性参考部分中的 AWS::CloudFront::Distribution 源。

API

要使用 AWS 开发工具包或 AWS Command Line Interface (AWS CLI) 对 CloudFront API 启用 Origin Shield，请使用 `OriginShield` 类型。您可以在 `OriginShield` 的 `Origin` 中指定 `DistributionConfig`。有关 `OriginShield` 类型的信息，请参阅《Amazon CloudFront API 参考》中的以下信息。

- OriginShield（类型）
- Origin（类型）
- DistributionConfig（类型）
- UpdateDistribution（操作）
- CreateDistribution（操作）

使用这些类型和操作的具体语法因开发工具包、CLI 或 API 客户端而异。有关更多信息，请参阅开发工具包、CLI 或客户端的参考文档。

估算 Origin Shield 成本

您可以根据作为增量层转到 Origin Shield 的请求数量来计算 Origin Shield 的费用。

对于通过代理到达源的动态（不可缓存）请求，Origin Shield 始终是增量层。动态请求使用 HTTP 方法：PUT、POST、PATCH 和 DELETE。

生存时间（TTL）设置小于 3600 秒的 GET 和 HEAD 请求被视为动态请求。此外，已禁用缓存的 GET 和 HEAD 请求也被视为动态请求。

要针对动态请求估计 Origin Shield 的费用，请使用以下公式：

动态请求总数 x 每 10,000 个请求的 Origin Shield 费用 / 10,000

对于采用 HTTP 方法 GET、HEAD 和 OPTIONS 的非动态请求，Origin Shield 有时是一个增量层。启用 Origin Shield 后，可以为 Origin Shield 选择 AWS 区域。对于自然转到与 Origin Shield 位于相同区域中的区域边缘缓存的请求，Origin Shield 不是增量层。您不会为这些请求累积 Origin Shield 费用。对于转到与 Origin Shield 位于不同区域中的区域边缘缓存，然后转到 Origin Shield 的请求，Origin Shield 为增量层。您确实需要为这些请求累积 Origin Shield 费用。

要针对可缓存的请求估计 Origin Shield 的费用，请使用以下公式：

可缓存请求总数 x（1 - 缓存命中率）x 从不同区域的区域边缘缓存转到 Origin Shield 的请求的百分比 x 每 10,000 个请求的 Origin Shield 费用 / 10,000

有关 Origin Shield 每 10,000 个请求收取的费用的更多信息，请参阅 CloudFront 定价。

Origin Shield 高可用性

Origin Shield 利用 CloudFront 区域边缘缓存功能。其中每个边缘缓存都在一个 AWS 区域中使用至少三个可用区以及自动伸缩 Amazon EC2 实例队列而构建。如果 Origin Shield 主位置不可用，则从 CloudFront 位置到 Origin Shield 的连接还会对每个请求使用活动错误跟踪，从而将请求自动路由到 Origin Shield 备用位置。

Origin Shield 如何与其他 CloudFront 功能进行交互

以下各节介绍 Origin Shield 如何与其他 CloudFront 功能进行交互。

Origin Shield 和 CloudFront 日志记录

要查看 Origin Shield 何时处理了请求，必须启用以下选项之一：

- CloudFront 标准日志（访问日志）。免费提供标准日志。
- CloudFront 实时日志。使用实时日志会产生额外费用。请参阅 Amazon CloudFront 定价。

来自 Origin Shield 的缓存命中在 CloudFront 日志的 `OriginShieldHit` 字段中显示为 `x-edge-detailed-result-type`。Origin Shield 利用 Amazon CloudFront 的区域边缘缓存。如果请求从 CloudFront 边缘站点路由到充当 Origin Shield 的区域边缘缓存，则在日志中将其报告为 `Hit`，而不是报告为 `OriginShieldHit`。

Origin Shield 和源组

Origin Shield 与 CloudFront 源组兼容。由于 Origin Shield 是源的一个属性，因此，对于每个源，请求始终会通过 Origin Shield，即使源是源组的一部分也是如此。对于给定的请求，CloudFront 通过主源的 Origin Shield 将请求路由到源组中的主源。如果该请求失败（根据源组故障转移标准），CloudFront 通过辅助源的 Origin Shield 将请求路由到辅助源。

Origin Shield 和 Lambda@Edge

Origin Shield 不会影响 Lambda@Edge 函数的功能，但会影响运行这些函数的 AWS 区域。

当您将 Origin Shield 与 Lambda@Edge 一起使用时，面向源的触发器（源请求和源响应）会在启用 Origin Shield 的 AWS 区域中运行。如果 Origin Shield 主位置不可用并且 CloudFront 将请求路由到 Origin Shield 备用位置，则面向源的 Lambda@Edge 触发器也将转为使用备用 Origin Shield 位置。

面向查看器的触发器不受影响。

通过 CloudFront 源失效转移来优化高可用性

对于需要高可用性的场景，您可以使用源故障转移功能设置 CloudFront。要开始使用，您需要创建一个具有两个源的源组：一个主源和一个辅助源。如果主源不可用，或返回指示故障的特定 HTTP 响应状态代码，则 CloudFront 自动切换到辅助源。

要设置源故障转移，您必须具有一个至少包含两个源的分配。接下来，为包含两个源（将其中一个设置为主源）的分配创建一个源组。最后，创建或更新缓存行为以使用源组。

要查看用于设置源组和配置特定源故障转移选项的步骤，请参阅创建源组。

在为缓存行为配置源故障转移后，CloudFront 将针对查看器请求执行以下操作：

- 当出现缓存命中结果时，CloudFront 将返回请求的对象。
- 当出现缓存未命中时，CloudFront 将请求路由至源组中的主源。
- 当主源返回不是为故障转移配置的状态代码（如 HTTP 2xx 或 3xx 状态代码）时，CloudFront 将向查看器提供请求的对象。
- 发生以下任何情况时：
  - 主源返回您为故障转移配置的 HTTP 状态代码
  - CloudFront 无法连接到主源
  - 来自主源的响应需要太长时间（超时）

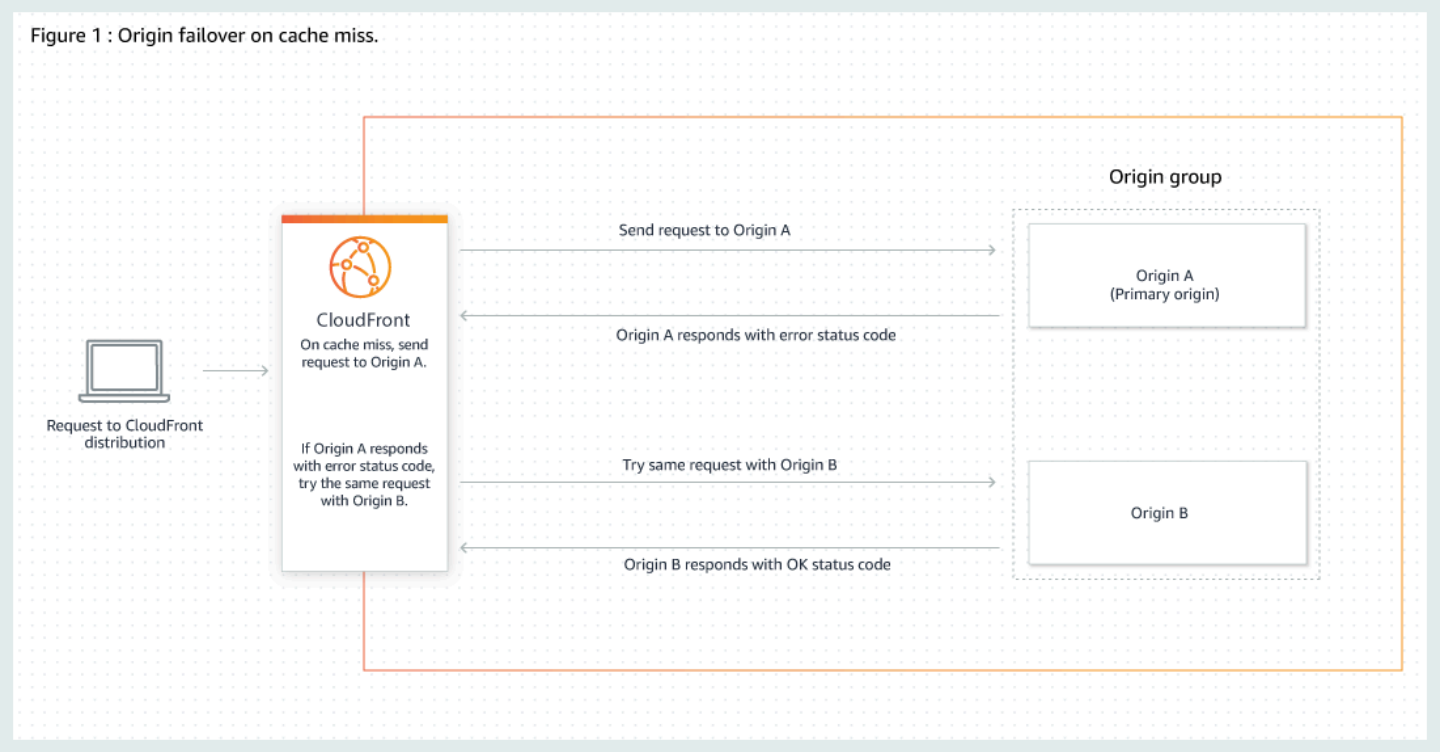
  然后，CloudFront 将请求路由到源组中的辅助源。

> **Note**
>
> 对于某些使用案例（如流视频内容），您可能希望 CloudFront 快速故障转移到辅助源。要调整 CloudFront 故障转移到辅助源的速度，请参阅控制源超时和尝试次数。

CloudFront 将所有传入的请求路由到主源，即使先前的请求故障转移到辅助源时也是如此。CloudFront 仅在向主源发送请求失败后才向辅助源发送请求。

只有当查看器请求的 HTTP 方法是 `GET`、`HEAD` 或 `OPTIONS` 时，CloudFront 才会故障转移到辅助源。当查看器发送不同的 HTTP 方法（例如 `POST`、`PUT` 等）时，CloudFront 不会进行故障转移。

下图阐述了源故障转移的工作原理。



主题

- 创建源组
- 控制源超时和尝试次数
- 将源故障转移与 Lambda@Edge 函数结合使用
- 将自定义错误页与源故障转移结合使用

创建源组

创建源组

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择要为其创建源组的分配。
3. 选择源选项卡。
4. 确保此分配有多个源。否则请添加第二个源。
5. 在源选项卡的（源组窗格中，选择创建源组。
6. 选择源组的源。添加源后，使用箭头设置优先级，即哪个源是主源组，哪个源是辅助源。
7. 为此源组输入一个名称。

8. 选择要用作故障转移条件的 HTTP 状态代码。您可以选择以下状态代码的任意组合：400、403、404、416、500、502、503 或 504。当 CloudFront 收到具有您指定的状态代码之一的响应时，它会故障转移到辅助源。

> **Note**
>
> 只有当查看器请求的 HTTP 方法是 GET、HEAD 或 OPTIONS 时，CloudFront 才会故障转移到辅助源。当查看器发送不同的 HTTP 方法（例如 POST、PUT 等）时，CloudFront 不会进行故障转移。

9. 选择创建源组。

请务必将您的源组指定为分配缓存行为的源。有关更多信息，请参阅 名称。

控制源超时和尝试次数

默认情况下，CloudFront 长达 30 秒（尝试 3 次连接，每次 10 秒）尝试连接到源组中的主源，之后故障转移到辅助源。对于某些使用案例（如流视频内容），您可能希望 CloudFront 更快地故障转移到辅助源。您可以调整以下设置，以影响 CloudFront 故障转移到辅助源的速度。如果源是辅助源或不属于源组的源，则这些设置会影响 CloudFront 向查看器返回 HTTP 504 响应的速度。

要更快地进行故障转移，请指定更短的连接超时、更少的连接尝试次数，或者同时指定两者。对于自定义源（包括配置为 静态网站托管的 Amazon S3 存储桶源），您还可以调整源响应超时。

源连接超时

源连接超时设置影响 CloudFront 尝试建立到源的连接时等待的时间。默认情况下，CloudFront 等待 10 秒以建立连接，但您可以指定 1-10 秒（含）。有关更多信息，请参阅 连接超时。

源连接尝试次数

源连接尝试设置会影响 CloudFront 尝试连接到源的次数。默认情况下，CloudFront 尝试 3 次进行连接，但您可以指定 1-3（含）。有关更多信息，请参阅 连接尝试次数。

对于自定义源（包括配置为静态网站托管的 Amazon S3 存储桶），此设置还影响在源响应超时的情况下 CloudFront 尝试从源获取响应的次数。

源响应超时

> **Note**
>
> 这仅适用于自定义源。

源响应超时设置影响 CloudFront 从源接收响应（或接收完整响应）的等待时间。默认情况下，CloudFront 等待 30 秒，但您可以指定 1-60 秒（含）。有关更多信息，请参阅 响应超时（仅自定义源）。

如何更改这些设置

在 CloudFront 控制台中更改这些设置

- 对于新源或新分配，您可以在创建资源时指定这些值。
- 对于现有分配中的现有源，可在编辑源时指定这些值。

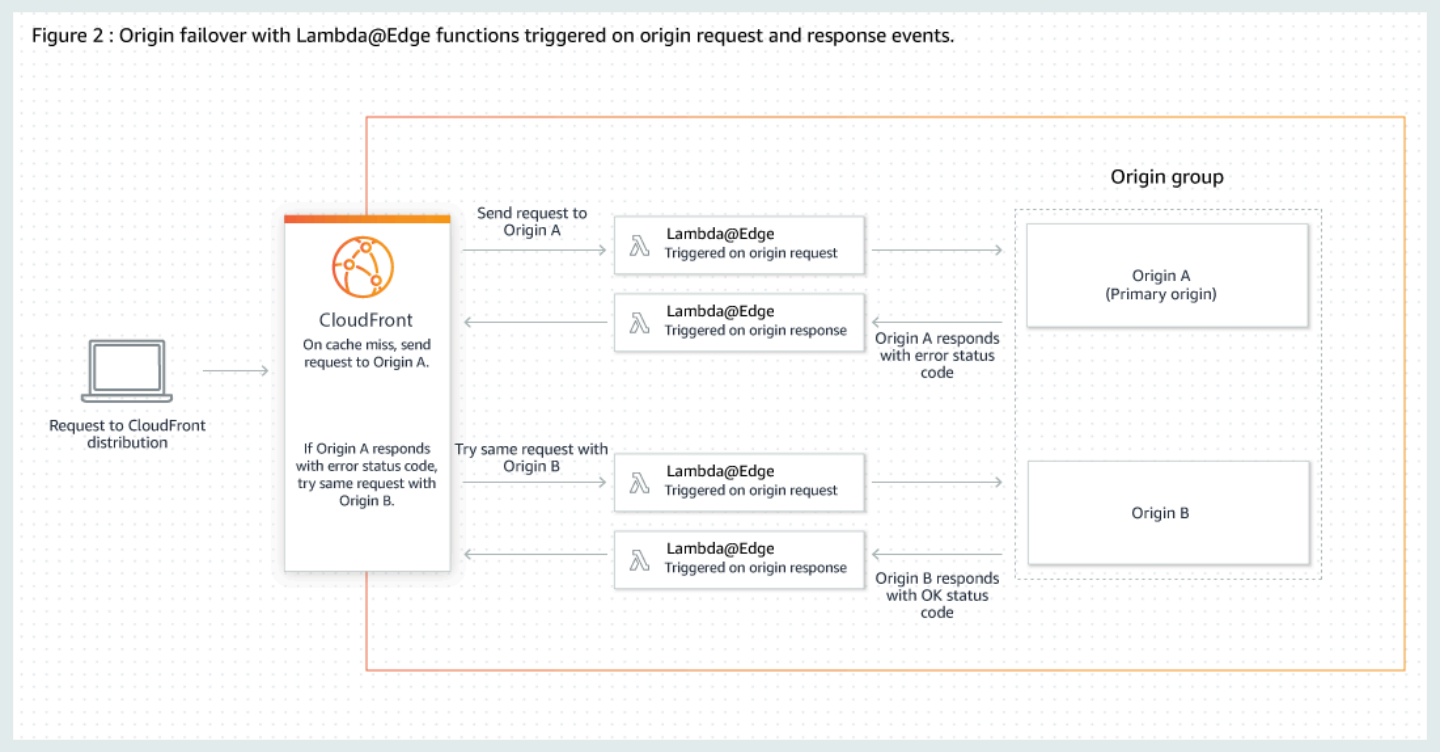
有关更多信息，请参阅 分配设置参考。

将源故障转移与 Lambda@Edge 函数结合使用

可以将 Lambda@Edge 函数与已使用源组设置的 CloudFront 分配结合使用。要使用 Lambda 函数，请在创建缓存行为时在源组的源请求或源响应触发器中指定它。当您将 Lambda@Edge 函数与源组一起使用时，对于单个查看器请求可以触发两次此函数。例如，考虑以下情景：

1. 使用源请求触发器创建 Lambda@Edge 函数。
2. 当 CloudFront（对于缓存未命中）向主源发送请求时会触发一次 Lambda 函数。
3. 主源使用为故障转移配置的 HTTP 状态代码进行响应。
4. 当 CloudFront 向辅助源发送相同的请求时，将再次触发 Lambda 函数。

下图阐述了源故障转移在源请求或响应触发器中包含 Lambda@Edge 函数时的工作方式。



有关如何使用 Lambda@Edge 触发器的更多信息，请参阅the section called “为 Lambda@Edge 函数添加触发器”。

有关管理 DNS 故障转移的更多信息，请参阅《Amazon Route 53 开发人员指南》中的配置 DNS 故障转移。

将自定义错误页与源故障转移结合使用

可以将自定义错误页面与源组结合使用，方式与将其与未为源故障转移设置的源结合使用的方式类似。

当使用源故障转移时，可将 CloudFront 配置为返回主源和/或辅助源的自定义错误页面：

- 返回主源的自定义错误页 – 如果主源返回未为故障转移配置的 HTTP 状态代码，则 CloudFront 将向查看器返回自定义错误页面。
- 返回辅助源的自定义错误页 – 如果 CloudFront 从辅助源收到故障状态代码，则 CloudFront 返回自定义错误页。

有关将自定义错误页面与 CloudFront 结合使用的更多信息，请参阅生成自定义错误响应。

管理内容保留在缓存中的时间长度（过期）

您可控制文件在 CloudFront 转发另一请求到您的源之前留在 CloudFront 缓存中的时长。减少持续时间让您可提供动态内容。增加持续时间意味着您的用户将获得更好的性能，因为直接从边缘缓存提供文件的可能性更大。较长的持续时间还会减少源的负载。

通常，CloudFront 从边缘站点提供文件，直至超出您指定的缓存时长，直至文件过期。文件过期后，边缘站点下次收到对文件的请求时，CloudFront 会将请求转发到源，以确认缓存包含文件的最新版本。来自源的响应取决于文件是否已更改：

- 如果 CloudFront 缓存已具有最新版本，则源将返回状态代码 `304 Not Modified`。
- 如果 CloudFront 缓存没有最新版本，则源将返回 `200 OK` 状态代码和文件的最新版本。

如果边缘站点中的某个文件不常被请求，则 CloudFront 可能会移除该文件 – 在文件到期日之前删除文件，以便为最近常被请求的文件腾出空间。

默认情况下，每个文件在 24 小时后自动过期，但您可以通过以下两种方式来更改默认行为：

- 要更改所有匹配相同路径模式的文件的缓存持续时间，可以更改缓存行为的最小 TTL、最大 TTL 和默认 TTL 的 CloudFront 设置。有关各个设置的信息，请参阅 the section called “分配设置” 中的最小 TTL、最大 TTL 和默认 TTL。
- 要更改单个文件的缓存持续时间，您可以配置源以向文件中添加 `Cache-Control` 标头以及 `max-age` 或 `s-maxage` 指令或者添加 `Expires` 标头。有关更多信息，请参阅 使用标头控制单独对象的缓存时间长度。

有关最小 TTL、默认 TTL 和最大 TTL 如何与 `max-age` 和 `s-maxage` 指令以及 `Expires` 标头字段交互的更多信息，请参阅the section called “指定 CloudFront 缓存对象的时间长度”。

您还可以控制在 CloudFront 将另一个请求转发到源以再次尝试获取请求的对象之前，在 CloudFront 缓存中保留错误（例如，`404 Not Found`）的时间。有关更多信息，请参阅 the section called “CloudFront 如何处理来自源的 HTTP 4xx 和 5xx 状态代码”。

主题

- 使用标头控制单独对象的缓存时间长度
- 提供过时（过期）的内容
- 指定 CloudFront 缓存对象的时间长度

- 使用 Amazon S3 控制台向对象添加标头

使用标头控制单独对象的缓存时间长度

您可以使用 Cache-Control 和 Expires 标头控制对象保留在缓存中的时间长度。最小 TTL、默认 TTL 和最大 TTL 的设置也会影响缓存时间长度，不过此处只简要说明标头对缓存时间长度的影响：

- Cache-Control max-age 指令让您指定希望对象在 CloudFront 再次从源服务器获取对象之前保留在缓存中的时长（以秒为单位）。CloudFront 支持的最短到期时间为 0 秒。最大值为 100 年。采用以下格式指定值：

  ```
  Cache-Control: max-age=#
  ```

  例如，以下指令告诉 CloudFront 在缓存中将相关对象保留 3600 秒（一小时）：

  ```
  Cache-Control: max-age=3600
  ```

  如果您希望对象在 CloudFront 边缘缓存中保留的时间不同于在浏览器缓存中的保留时间，可以将 Cache-Control max-age 和 Cache-Control s-maxage 指令一起使用。有关更多信息，请参阅 指定 CloudFront 缓存对象的时间长度。
- Expires标头字段让您使用 RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1 Section 3.3.1, Full Date 中指定的格式指定过期日期和时间，例如：

  ```
  Sat, 27 Jun 2015 23:59:59 GMT
  ```

建议您使用 Cache-Control max-age 指令代替 Expires 标头字段，以控制对象缓存。如果您同时指定 Cache-Control max-age 和 Expires 的值，CloudFront 则只使用 Cache-Control max-age 的值。

有关更多信息，请参阅 指定 CloudFront 缓存对象的时间长度。

您不能使用查看器的 Cache-Control 请求中的 HTTP Pragma 或 GET 标头字段来强制 CloudFront 返回到对象的源服务器。CloudFront 将忽略查看器请求中的这些标头字段。

有关 Cache-Control 和 Expires 标头字段的更多信息，请参阅《RFC 2616，超文本传输协议 – HTTP/1.1》中以下章节：

- 第 14.9 节：缓存控制
- 第 14.21 节：过期

提供过时（过期）的内容

CloudFront 支持 `Stale-While-Revalidate` 和 `Stale-If-Error` 缓存控制指令。

- `stale-while-revalidate` 指令允许 CloudFront 从缓存中提供过时内容，同时它从源异步获取新版本。这样可以缩短延迟，因为用户无需等待后台获取即可立即收到来自 CloudFront 边缘站点的响应，并且新内容会在后台加载以供将来的请求使用。

  在以下示例中，CloudFront 将响应缓存一个小时（`max-age=3600`）。如果在此时间段之后发出请求，CloudFront 将提供过时的内容，同时向源发送请求以重新验证和刷新缓存的内容。在重新验证内容期间，过时内容最多可提供 10 分钟（`stale-while-revalidate=600`）。

  ```
  Cache-Control: max-age=3600, stale-while-revalidate=600
  ```

- `stale-if-error` 指令允许 CloudFront 在源不可访问或源返回介于 500 和 600 之间的错误代码时从缓存中提供过时内容。这可确保查看器即使在源中断期间也可以访问内容。

  在以下示例中，CloudFront 将响应缓存一个小时（`max-age=3600`）。如果源在此时间段后断开或返回错误，CloudFront 将在长达 24 小时（`stale-if-error=86400`）内继续提供过时内容。

  ```
  Cache-Control: max-age=3600, stale-if-error=86400
  ```

  > **Note**
  >
  > 当配置了 `stale-if-error` 和自定义错误响应时，如果在指定的 `stale-if-error` 持续时间内遇到错误，CloudFront 将首先尝试提供过时的内容。如果过时内容不可用或内容已超过 `stale-if-error` 持续时间，CloudFront 会提供为相应错误状态码配置的自定义错误响应。

将两者一起使用

`stale-while-revalidate` 和 `stale-if-error` 是独立的缓存控制指令，可以一起使用以减少延迟并为您的源添加缓冲区以进行响应或恢复。

在以下示例中，CloudFront 将响应缓存一个小时（`max-age=3600`）。如果在此时间段之后发出请求，CloudFront 将在重新验证内容期间提供最长 10 分钟（`stale-while-revalidate=600`）的过时内容。如果在 CloudFront 尝试重新验证内容时原始服务器返回错误，CloudFront 将在最长 24 小时（`stale-if-error=86400`）内继续提供过时内容。

```
Cache-Control: max-age=3600, stale-while-revalidate=600, stale-if-error=86400
```

> **Tip**
>
> 缓存是性能和新鲜度之间的平衡。使用 `stale-while-revalidate` 和 `stale-if-error` 之类的指令可以增强性能和用户体验，但要确保配置与您所需的内容新鲜度相一致。过时内容指令最适合需要刷新内容但并非必须具有最新版本的使用案例。此外，如果您的内容没有更改或极少更改，`stale-while-revalidate` 可能会增加不必要的网络请求。相反，可以考虑设置较长的缓存持续时间。

指定 CloudFront 缓存对象的时间长度

要控制 CloudFront 在将另一个请求发送到源之前在缓存中保留对象的时长，您可以：

- 设置 CloudFront 分配的缓存行为中的最小、最大和原定设置 TTL 值。您可以在附加到缓存行为的缓存策略中设置这些值（推荐），或在原有缓存设置中进行设置。
- 将 `Cache-Control` 或 `Expires` 标头包含在来自源的响应中。这些标头还有助于确定浏览器在浏览器缓存中保留对象的时间长度，超过该时间长度后才会将另一个请求发送到 CloudFront。

下表说明了从源发送的 `Cache-Control` 和 `Expires` 标头如何与缓存行为中的 TTL 设置配合使用，从而影响缓存的。

源标头	最小 TTL = 0	最小 TTL > 0
源将 **`Cache-Control: max-age`** 指令添加到对象	CloudFront 缓存 CloudFront 缓存对象的时间长度为 `Cache-Control: max-age` 指令或 CloudFront 最长 TTL 的值（取二者中的较小值）。 浏览器缓存	CloudFront 缓存 CloudFront 缓存取决于 CloudFront 最短 TTL 和最长 TTL 的值以及 `Cache-Control max-age` 指令： • 如果最短 TTL < `max-age` < 最长 TTL，则 CloudFront 缓存对象的时间长度为

源标头	最小 TTL = 0	最小 TTL > 0
	浏览器缓存对象的时间长度为 `Cache-Control: max-age` 指令的值。	`Cache-Control: max-age` 指令的值。 • 如果 `max-age` < 最短 TTL，则 CloudFront 缓存对象的时间长度为 CloudFront 最短 TTL 的值。 • 如果 `max-age` > 最长 TTL，则 CloudFront 缓存对象的时间长度为 CloudFront 最长 TTL 的值。 浏览器缓存 浏览器缓存对象的时间长度为 `Cache-Control: max-age` 指令的值。
源不会将 **`Cache-Control: max-age`** 指令添加到对象	CloudFront 缓存 CloudFront 缓存对象的时间长度为 CloudFront 默认 TTL 值。 浏览器缓存 视浏览器而定。	CloudFront 缓存 CloudFront 缓存对象的时间长度为 CloudFront 最小 TTL 或默认 TTL（以较大的值为准）。 浏览器缓存 视浏览器而定。

源标头	最小 TTL = 0	最小 TTL > 0
源将 **Cache-Control: max-age** 和 **Cache-Control: s-maxage** 指令添加到对象	CloudFront 缓存 CloudFront 缓存对象的时间长度为 Cache-Control: s-maxage 指令或 CloudFront 最长 TTL 的值（取二者中的较小值）。 浏览器缓存 浏览器缓存对象的时间长度为 Cache-Control max-age 指令的值。	CloudFront 缓存 CloudFront 缓存取决于 CloudFront 最短 TTL 和最长 TTL 的值以及 Cache-Control: s-maxage 指令： • 如果最短 TTL < s-maxage < 最长 TTL，则 CloudFront 缓存对象的时间长度为 Cache-Control: s-maxage 指令的值。 • 如果 s-maxage < 最短 TTL，则 CloudFront 缓存对象的时间长度为 CloudFront 最短 TTL 的值。 • 如果 s-maxage > 最长 TTL，则 CloudFront 缓存对象的时间长度为 CloudFront 最长 TTL 的值。 浏览器缓存 浏览器缓存对象的时间长度为 Cache-Control: max-age 指令的值。

源标头	最小 TTL = 0	最小 TTL > 0
源将 **Expires** 标头添加到对象	CloudFront 缓存 CloudFront 缓存对象，直至 `Expires` 标头中的日期或 CloudFront 最长 TTL 的值，以先到者为准。 浏览器缓存 浏览器缓存对象，直至 `Expires` 标头中的日期。	CloudFront 缓存 CloudFront 缓存取决于 CloudFront 最短 TTL 和最长 TTL 以及 `Expires` 标头的值： • 如果最短 TTL < `Expires` < 最长 TTL，则 CloudFront 会缓存对象至 `Expires` 标头中的日期和时间。 • 如果 `Expires` < 最短 TTL，则 CloudFront 缓存对象的时间长度为 CloudFront 最短 TTL 的值。 • 如果 `Expires` > 最长 TTL，则 CloudFront 缓存对象的时间长度为 CloudFront 最长 TTL 的值。 浏览器缓存 浏览器缓存对象，直至 `Expires` 标头中的日期和时间。

源标头	最小 TTL = 0	最小 TTL > 0
源将 **Cache-Control: no-cache、no-store** 和/或 **private** 指令添加到对象	CloudFront 和浏览器以标头为准。	CloudFront 缓存 CloudFront 缓存对象的时间长度为 CloudFront 最短 TTL 值。请参阅此表下面的警告。 浏览器缓存 浏览器以标头为准。

> **Warning**
>
> 如果您的最小 TTL 大于 0，CloudFront 将使用缓存策略的最小 TTL，即使源标头中存在 `Cache-Control: no-cache、no-store` 和/或 `private` 指令。
> 如果源可以访问，CloudFront 会从源获取对象并将其返回给查看器。
> 如果源无法访问并且最小或 最大 TTL 值大于 0，CloudFront 将提供其先前从源获取的对象。为了避免该行为，请将 `Cache-Control: stale-if-error=0` 指令包含在从源返回的对象中。这样一来，如果源无法访问，以后 CloudFront 就会返回错误消息，作为对请求的响应，而不是返回其先前从源获取的对象。

有关如何使用 CloudFront 控制台更改分配的设置的信息，请参阅更新分配。有关如何使用 CloudFront API 更改分配的设置的信息，请参阅 UpdateDistribution。

使用 Amazon S3 控制台向对象添加标头

使用 Amazon S3 控制台向 Amazon S3 对象添加 **Cache-Control** 或 **Expires** 标头字段

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在存储桶列表中，选择包含要向其中添加标头的文件的存储桶的名称。
3. 选中要向其中添加标头的文件或文件夹名称旁边的复选框。将标头添加到文件夹时，它会影响该文件夹内的所有文件。
4. 选择操作，然后选择编辑元数据。

5. 在添加元数据面板中，执行以下操作：

   a. 选择添加元数据。

   b. 对于类型，选择系统定义。

   c. 对于键，选择要添加的标头的名称（Cache-Control 或 Expires）。

   d. 对于值，输入标头值。例如，对于 `Cache-Control` 标头，您可以输入 `max-age=86400`。对于 `Expires`，您可以输入到期日期和时间，如 `Wed, 30 Jun 2021 09:28:00 GMT`。

6. 在页面底部，选择编辑元数据。

根据查询字符串参数缓存内容

一些 Web 应用程序使用查询字符串将信息发送到源。查询字符串是 Web 请求的一部分，显示在 ? 字符之后；该字符串可以包含一个或多个使用 & 字符分隔的参数。在以下示例中，查询字符串包括两个参数 *color=red* 和 *size=large*：

`https://d111111abcdef8.cloudfront.net/images/image.jpg?`*`color=red&size=large`*

对于分配，您可选择是否希望 CloudFront 将查询字符串转发到源，以及是基于所有参数还是基于选定参数缓存内容。为什么说这可能很有用？ 考虑以下 示例。

假设您的网站提供五种语言。网站的全部五个版本的目录结构和文件名均相同。用户查看网站时，转发到 CloudFront 的请求包括语言查询字符串参数，该参数基于用户选择的语言。您可以将 CloudFront 配置为将查询字符串转发到源，并基于语言参数进行缓存。如果您将 Web 服务器配置为返回指定页面的与所选语言对应的版本，CloudFront 将基于语言查询字符串参数的值分别缓存各个语言版本。

在此示例中，如果您网站的主页为 `main.html`，则以下五个请求将导致 CloudFront 缓存 `main.html` 五次，为语言查询字符串参数的每个值缓存一次：

- `https://d111111abcdef8.cloudfront.net/main.html?`*`language=de`*
- `https://d111111abcdef8.cloudfront.net/main.html?`*`language=en`*
- `https://d111111abcdef8.cloudfront.net/main.html?`*`language=es`*
- `https://d111111abcdef8.cloudfront.net/main.html?`*`language=fr`*
- `https://d111111abcdef8.cloudfront.net/main.html?`*`language=jp`*

请注意以下几点：

- 一些 HTTP 服务器不处理查询字符串参数，因此，不基于参数值返回对象的不同版本。对于这些源，如果将 CloudFront 配置为将查询字符串参数转发到源，CloudFront 仍根据参数值进行缓存，即使源针对每个参数值向 CloudFront 返回完全相同的对象版本。
- 要使查询字符串参数与语言结合使用（如上述示例中所述），您必须使用 & 字符作为查询字符串参数之间的分隔符。如果您使用不同的分隔符，则可能获得意外结果，具体取决于您为 CloudFront 指定的用作缓存基础的参数以及这些参数在查询字符串中的显示顺序。

  以下示例说明在您使用其他分隔符并将 CloudFront 配置为仅根据 `color` 参数进行缓存时发生的情况：

  - 在以下请求中，CloudFront 根据 `color` 参数值缓存您的内容，但 CloudFront 将值解释为 *`red;size=large`*：

    ```
    https://d111111abcdef8.cloudfront.net/images/
    image.jpg?color=red;size=large
    ```

  - 在以下请求中，CloudFront 缓存内容，但不根据查询字符串参数进行缓存。这是因为您将 CloudFront 配置成了基于 `color` 参数进行缓存，但 CloudFront 将以下字符串解释为仅包含 `size` 参数且其值为 *`large;color=red`*：

    ```
    https://d111111abcdef8.cloudfront.net/images/
    image.jpg?size=large;color=red
    ```

您可以将 CloudFront 配置为执行下列操作之一：

- 完全不将查询字符串转发到源。如果您不转发查询字符串，CloudFront 不基于查询字符串参数进行缓存。
- 将查询字符串转发到源，并基于查询字符串中的所有参数进行缓存。
- 将查询字符串转发到源，并基于查询字符串中的指定参数进行缓存。

有关更多信息，请参阅 the section called “优化缓存”。

主题

- 针对查询字符串转发和缓存的控制台和 API 设置
- 优化缓存
- 查询字符串参数和 CloudFront 标准日志（访问日志）

针对查询字符串转发和缓存的控制台和 API 设置

要在 CloudFront 控制台中配置查询字符串转发和缓存，请参阅 the section called “分配设置” 中的以下设置：

- the section called “查询字符串转发和缓存”
- the section called “查询字符串允许列表”

要使用 CloudFront API 配置查询字符串转发和缓存，请参阅《Amazon CloudFront API 参考》中 DistributionConfig 和 DistributionConfigWithTags 的以下设置：

- `QueryString`
- `QueryStringCacheKeys`

优化缓存

在将 CloudFront 配置为根据查询字符串参数进行缓存时，可以执行以下步骤来减少 CloudFront 转发到源的请求数。当 CloudFront 边缘站点为对象提供服务时，您可以减少源服务器上的负载并减少延迟，因为从更接近用户的位置为对象提供了服务。

仅基于源为其返回不同版本的对象的参数进行缓存

对于您的 Web 应用程序转发到 CloudFront 的每个查询字符串参数，CloudFront 针对每个参数值将请求转发到您的源，并为每个参数值缓存对象的单独版本。即使不论参数值如何，源始终返回相同的对象时，也是如此。对于多个参数，请求数和对象数相乘。

建议您将 CloudFront 配置为仅基于源会返回不同版本的查询字符串参数进行缓存，并建议您谨慎考虑基于各参数进行缓存的优点。例如，假设您有一个零售网站。您有六种不同颜色的夹克衫的照片，夹克衫有 10 种不同的尺寸。您的夹克衫图片会显示不同颜色，但没有不同尺寸。要优化缓存，您应将 CloudFront 配置为仅基于颜色参数进行缓存，而非尺寸参数。这增加了 CloudFront 可从缓存满足请求的可能性，这提高了性能并降低了源的负载。

始终按相同的顺序列出参数

查询字符串中参数的顺序很重要。在以下示例中，查询字符串相同，但参数的顺序不同。这将导致 CloudFront 将两个针对 image.jpg 的单独请求转发到源，并缓存对象的两个单独版本：

- `https://d111111abcdef8.cloudfront.net/images/image.jpg?color=red&size=large`

- https://d111111abcdef8.cloudfront.net/images/image.jpg?*size=large&color=red*

建议您始终按相同的顺序列出参数名称，例如字母顺序。

始终为参数名称和值使用相同的大小写

CloudFront 在基于查询字符串参数进行缓存时，会考虑参数名称和值的大小写情况。在以下示例中，查询字符串相同，但参数名称和值的大小写不同。这将导致 CloudFront 将四个针对 image.jpg 的单独请求转发到源，并缓存对象的四个单独版本：

- https://d111111abcdef8.cloudfront.net/images/image.jpg?*color=red*
- https://d111111abcdef8.cloudfront.net/images/image.jpg?*color=Red*
- https://d111111abcdef8.cloudfront.net/images/image.jpg?*Color=red*
- https://d111111abcdef8.cloudfront.net/images/image.jpg?*Color=Red*

建议您为参数名称和值使用一致的大小写，例如全小写。

不要使用与签名 URL 冲突的参数名称

如果您使用签名 URL 限制访问您的内容（如果您添加了可信签署人到您的分配），CloudFront 在转发 URL 剩余部分到您的源之前将删除以下查询字符串参数：

- Expires
- Key-Pair-Id
- Policy
- Signature

如果您使用已签名 URL 并且希望将 CloudFront 配置为将查询字符串转发到源，您自己的查询字符串参数不能命名为 Expires、Key-Pair-Id、Policy 或 Signature。

查询字符串参数和 CloudFront 标准日志（访问日志）

如果您启用日志记录，则 CloudFront 会记录完整的 URL，包括查询字符串参数。无论您是否将 CloudFront 配置为将查询字符串转发到源都是如此。有关 CloudFront 日志记录的更多信息，请参阅the section called “使用标准日志（访问日志）”。

根据 Cookie 缓存内容

默认情况下，CloudFront 在处理请求和响应时或在边缘站点中缓存对象时，不考虑 Cookie。如果 CloudFront 收到两个内容相同（仅 `Cookie` 标头中的内容不同）的请求，默认情况下，CloudFront 会将这两个请求视为相同并为它们返回相同的对象。

您可以将 CloudFront 配置为将查看器请求中的部分或所有 Cookie 转发到您的源，以及基于转发的 Cookie 值缓存对象的单独版本。在执行此操作时，CloudFront 将使用查看器请求中的部分或所有 Cookie（无论配置为转发哪些 Cookie）来唯一标识缓存中的对象。

例如，假设 `locations.html` 的请求包含 `country` Cookie，其值为 `uk` 或 `fr`。在将 CloudFront 配置为根据 `country` Cookie 的值缓存对象时，CloudFront 会将 `locations.html` 请求转发到源并包含 `country` Cookie 及其值。源返回 `locations.html`，并且 CloudFront 为具有 `country` Cookie 值 `uk` 的请求缓存对象一次，为具有 Cookie 值 `fr` 的请求缓存一次。

> **Important**
>
> Amazon S3 和部分 HTTP 服务器不处理 Cookie。请不要将 CloudFront 配置为将 Cookie 转发到不处理 Cookie 或不根据 Cookie 改变其响应的源。这可能会导致 CloudFront 将更多请求转发到同一对象的源，从而降低性能并增加源上的负载。考虑到上一个示例，如果您的源不处理 `country` Cookie，或者始终将相同版本的 `locations.html` 返回到 CloudFront，而不考虑 `country` Cookie 的值，请不要将 CloudFront 配置为转发该 Cookie。
>
> 相反，如果您的自定义源依赖于某个特定的 Cookie，或者根据 Cookie 发送不同的响应，请确保将 CloudFront 配置为将该 Cookie 转发到源。否则，CloudFront 会在将请求转发到您的源之前删除该 Cookie。

要配置 Cookie 转发，您需要更新分配的缓存行为。有关缓存行为的更多信息，请参阅缓存行为设置，特别是转发 Cookie 和允许列表 Cookie部分。

可以配置每个缓存行为以执行下列操作之一：

- 将所有 Cookie 转发到您的源 – CloudFront 包含查看器在将请求转发到源时发送的所有 Cookie。当源返回响应时，CloudFront 会使用查看器请求中的 Cookie 名称和值来缓存响应。如果源响应包含 `Set-Cookie` 标头，CloudFront 会将其与请求的对象一起返回给查看器。CloudFront 还与从源返回的对象一起缓存 `Set-Cookie` 标头，并在所有缓存命中时将这些 `Set-Cookie` 标头发送给查看器。

- 转发您指定的 Cookie 集 – CloudFront 将删除查看器发送的未列入允许名单的所有 Cookie，然后再将请求转发到源。CloudFront 使用查看器请求中列出的 cookie 名称和值缓存响应。如果源响应包含 `Set-Cookie` 标头，CloudFront 会将其与请求的对象一起返回给查看器。CloudFront 还与从源返回的对象一起缓存 `Set-Cookie` 标头，并在所有缓存命中时将这些 `Set-Cookie` 标头发送给查看器。

  有关在 Cookie 名称中指定通配符的信息，请参阅允许列表 Cookie。

  如需了解您可以为每个缓存行为转发的 Cookie 名称的当前数量配额或需要请求提高配额，请参阅查询字符串的配额（旧缓存设置）。
- 不将 Cookie 转发到源 – CloudFront 不根据查看器发送的 Cookie 缓存您的对象。此外，CloudFront 会先删除 Cookie，然后再将请求转发到您的源，并且会在将响应返回给您的查看器之前从响应中删除 `Set-Cookie` 标头。由于这不是使用源资源的最佳方式，因此当您选择此缓存行为时，应确保您的源默认情况下不在源响应中包含 Cookie。

请注意以下有关指定要转发的 Cookie 的信息：

访问日志

如果将 CloudFront 配置为记录请求和 Cookie，则 CloudFront 会记录所有 Cookie 以及所有 Cookie 属性，即使将 CloudFront 配置为不将 Cookie 转发到源，或者将 CloudFront 配置为仅转发特定的 Cookie。有关 CloudFront 日志记录的更多信息，请参阅配置和使用标准日志（访问日志）。

区分大小写

Cookie 的名称和值都要区分大小写。例如，如果将 CloudFront 配置为转发所有 Cookie，并且同一对象的两个查看器请求具有相同的 Cookie（仅大小写不同），则 CloudFront 会将该对象缓存两次。

CloudFront 对 cookie 进行排序

如果将 CloudFront 配置为转发 Cookie（全部或某个子集），则 CloudFront 会在将请求转发到您的源之前以自然顺序按 Cookie 名称对 Cookie 进行排序。

`If-Modified-Since` 和 `If-None-Match`

如果将 CloudFront 配置为转发 Cookie（全部或某个子集），则不支持 `If-Modified-Since` 和 `If-None-Match` 条件请求。

需要标准名称/值对格式

CloudFront 仅在值遵循标准名称/值对格式时转发 Cookie 标头，例如："Cookie: cookie1=value1; cookie2=value2"

禁止对 **Set-Cookie** 标头进行缓存

如果将 CloudFront 配置为将 Cookie 转发到源（全部或特定的 Cookie），它还会缓存源响应中收到的 Set-Cookie 标头。CloudFront 在其对原始查看器的响应中包括这些 Set-Cookie 标头，并将它们包含在从 CloudFront 缓存提供的后续响应中。

如果您希望在源处接收 Cookie，但不希望 CloudFront 在源的响应中缓存 Set-Cookie 标头，请将源配置为使用指定 Cache-Control 作为字段名的 no-cache 指令来添加 Set-Cookie 标头。例如：Cache-Control: no-cache="Set-Cookie"。有关更多信息，请参阅超文本传输协议 (HTTP/1.1)：缓存 标准中的响应缓存控制指令。

Cookie 名称的最大长度

如果将 CloudFront 配置为将特定的 Cookie 转发到源，则配置为由 CloudFront 转发的所有 Cookie 名称的总字节数不能超过 512 与您转发的 Cookie 数的差。例如，如果您将 CloudFront 配置为将 10 个 Cookie 转发到源，则 10 个 Cookie 的名称组合长度不能超过 502 个字节 (512 - 10)。

如果您将 CloudFront 配置为将所有 Cookie 转发到源，Cookie 名称的长度没有关系。

有关使用 CloudFront 控制台更新分配以使 CloudFront 将 Cookie 转发到源的信息，请参阅更新分配。有关使用 CloudFront API 更新分配的信息，请参阅《Amazon CloudFront API 参考》中的 UpdateDistribution。

根据请求标头缓存内容

CloudFront 允许您选择是否希望 CloudFront 将标头转发到源并根据查看器请求中的标头值缓存指定对象的不同版本。这样，您便可以根据用户使用的设备、查看器的位置、查看器使用的语言及各种其他条件来提供内容的不同版本。

主题

- 标头和分配 – 概述
- 选择缓存所基于的标头
- 将 CloudFront 配置为遵守 CORS 设置
- 配置基于设备类型的缓存

- 配置基于查看器语言的缓存
- 配置基于查看器位置的缓存
- 配置基于请求协议的缓存
- 为压缩文件配置缓存
- 根据标头进行缓存如何影响性能
- 标头和标头值的大小写如何影响缓存
- CloudFront 返回给查看器的标头

标头和分配 – 概述

默认情况下，CloudFront 在边缘站点中缓存对象时不考虑标头。如果源返回两个对象并且这两个对象仅有请求标头中的值不同，CloudFront 仅缓存对象的一个版本。

您可以将 CloudFront 配置为将标头转发到源，这会导致 CloudFront 根据一个或多个请求标头中的值缓存某个对象的多个版本。要根据特定标头的值配置 CloudFront 以缓存对象，请为分配指定缓存行为设置。有关更多信息，请参阅基于选择的请求标头进行缓存。

例如，假设 `logo.jpg` 的查看器请求包含自定义 `Product` 标头，其值为 `Acme` 或 `Apex`。当您将 CloudFront 配置为基于 `Product` 标头的值缓存对象时，CloudFront 将针对 `logo.jpg` 的请求转发到源并包括 `Product` 标头和标头值。CloudFront 为 `logo.jpg` 标头中值为 `Product` 的请求缓存 `Acme` 一次，为其中值为 `Apex` 的请求缓存一次。

可以在分配中配置每个缓存行为，以执行以下操作之一：

- 将所有标头转发到源

> **Note**
>
> 对于旧缓存设置 - 如果您将 CloudFront 配置为将所有标头转发到源，则 CloudFront 不会缓存与此缓存行为关联的对象。相反，它会将每个请求发送到源。

- 转发您指定的标头列表。CloudFront 会基于所有指定标头中的值缓存对象。CloudFront 默认还会转发其转发的标头，但它仅基于指定的标头缓存对象。
- 仅转发默认标头。在此配置中，CloudFront 不会基于请求标头中的值缓存您的对象。

如需了解您可以为每个缓存行为转发的标头的当前数量配额或请求提高配额，请参阅标头的配额。

有关使用 CloudFront 控制台更新分配以使 CloudFront 将标头转发到源的信息，请参阅更新分配。有关使用 CloudFront API 更新现有分配的信息，请参阅《Amazon CloudFront API 参考》中的 UpdateDistribution。

选择缓存所基于的标头

您可以转发到源以及 CloudFront 进行缓存所基于的标头取决于您的源是 Amazon S3 存储桶还是自定义源。

- Amazon S3 – 您可以将 CloudFront 配置为根据特定标头的数量缓存您的对象（请参阅下面的例外情况列表）。但是，建议您避免使用 Amazon S3 源转发标头，除非您需要实施跨源资源共享（CORS），或者想要通过在面向源的事件中使用 Lambda@Edge 来对内容进行个性化设置。
  - 要配置 CORS，您必须转发允许 CloudFront 为启用了跨源资源共享 (CORS) 的网站分配内容的标头。有关更多信息，请参阅 将 CloudFront 配置为遵守 CORS 设置。
  - 要通过使用转发至您的 Amazon S3 源的标头对内容进行个性化设置，请编写和添加 Lambda@Edge 函数，并将这些函数与要由某个面向源的事件触发的 CloudFront 分配相关联。有关使用标头对内容进行个性化设置的更多信息，请参阅按国家/地区或设备类型标头个性化内容 - 示例。

    建议您避免转发并非用于对内容进行个性化设置的标头，因为转发额外的标头可能会降低您的缓存命中率。即，CloudFront 无法从边缘缓存来服务与所有请求中占的比例一样多的请求。
- 自定义源 – 您可以将 CloudFront 配置为根据以下项以外的任意请求标头的值进行缓存：
  - `Connection`
  - `Cookie` – 如果希望根据 Cookie 转发和缓存，您可以在分配中使用单独的设置。有关更多信息，请参阅 根据 Cookie 缓存内容。
  - `Host (for Amazon S3 origins)`
  - `Proxy-Authorization`
  - `TE`
  - `Upgrade`

  您可以将 CloudFront 配置为基于 `Date` 和 `User-Agent` 标头中的值缓存对象，但建议您不要这样做。这些标头具有大量可能的值，并且基于其值的缓存操作会导致 CloudFront 将更多请求转发到源。

有关 HTTP 请求标头的完整列表以及 CloudFront 如何处理这些标头，请参阅HTTP 请求标头和 CloudFront 行为（自定义源和 Amazon S3 源）。

将 CloudFront 配置为遵守 CORS 设置

如果您已在 Amazon S3 存储桶或自定义源上启用跨源资源共享 (CORS)，则必须选择要转发的特定标头以遵守 CORS 设置。您必须转发的标头因源（Amazon S3 或自定义）及您是否要缓存 OPTIONS 响应而异。

Amazon S3

- 如果要缓存 OPTIONS 响应，请执行以下操作：
  - 为默认缓存行为设置选择启用 OPTIONS 响应缓存的选项。
  - 配置 CloudFront 以转发以下标头：`Origin`、`Access-Control-Request-Headers`、和 `Access-Control-Request-Method`。
- 如果您不希望缓存 OPTIONS 响应，请将 CloudFront 配置为转发 `Origin` 标头以及源所需的任何其他标头（例如 `Access-Control-Request-Headers`、`Access-Control-Request-Method` 或其他）。

自定义源 – 转发 `Origin` 标头以及源所需的任何其他标头。

要将 CloudFront 配置为根据 CORS 缓存响应，必须将 CloudFront 配置为使用缓存策略转发标头。有关更多信息，请参阅 使用策略来控制缓存键。

有关 CORS 和 Amazon S3 的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的启用跨源资源共享（CORS）。

配置基于设备类型的缓存

如果您希望 CloudFront 基于用户查看内容所用设备来缓存不同版本的对象，可将 CloudFront 配置为将合适的标头转发给您的自定义源：

- `CloudFront-Is-Desktop-Viewer`
- `CloudFront-Is-Mobile-Viewer`
- `CloudFront-Is-SmartTV-Viewer`
- `CloudFront-Is-Tablet-Viewer`

根据 `User-Agent` 标头的值，在将请求转发给您的源之前，CloudFront 将这些标头的值设置为 `true` 或 `false`。如果某个设备归入多个类别中，则多个值可能为 `true`。例如，对于一些平板电脑设

备，CloudFront 可能将 CloudFront-Is-Mobile-Viewer 和 CloudFront-Is-Tablet-Viewer 设置为 true。

配置基于查看器语言的缓存

如果您希望 CloudFront 根据请求中指定的语言缓存对象的不同版本，请将 CloudFront 配置为将 Accept-Language 标头转发到源。

配置基于查看器位置的缓存

如果您希望 CloudFront 根据发出请求的国家/地区缓存对象的不同版本，请将 CloudFront 配置为将 CloudFront-Viewer-Country 标头转发到源。CloudFront 自动将发出请求的 IP 地址转换为两个字母的国家/地区代码。要获取可按代码和国家/地区名称排序的易于使用的国家/地区代码列表，请参阅 Wikipedia 条目 ISO 3166-1 alpha-2。

配置基于请求协议的缓存

如果您希望 CloudFront 根据请求的协议（HTTP 或 HTTPS）缓存对象的不同版本，请配置 CloudFront 以将 CloudFront-Forwarded-Proto 标头转发到源。

为压缩文件配置缓存

如果源支持 Brotli 压缩，您可以基于 Accept-Encoding 标头进行缓存。仅当源根据 Accept-Encoding 标头处理不同内容时基于该标头配置缓存。

根据标头进行缓存如何影响性能

在您将 CloudFront 配置为基于一个或多个标头进行缓存并且标头有多个可能值时，CloudFront 会为同一个对象将多个请求转发到您的源服务器。这会降低性能并增加源服务器上的负载。如果不管指定标头的值如何，源服务器均返回相同的对象，建议您不要将 CloudFront 配置为基于该标头进行缓存。

如果您将 CloudFront 配置为转发多个标头，只要值相同，查看器请求中的标头顺序对缓存没有影响。例如，如果一个请求包含标头 A:1,B:2，而另一个请求包含 B:2,A:1，CloudFront 只缓存对象的一个副本。

标头和标头值的大小写如何影响缓存

根据标头值进行缓存时，CloudFront 不会考虑标头名称的大小写，但是会考虑标头值的大小写：

- 如果查看器请求同时包含 `Product:Acme` 和 `product:Acme`，CloudFront 仅缓存对象一次。它们之间唯一的差别是标头名称的大小写，这不会影响缓存。
- 如果查看器请求同时包括 `Product:Acme` 和 `Product:acme`，CloudFront 缓存对象两次，因为在一些请求中具有值 `Acme`，而在另一些请求中具有值 `acme`。

CloudFront 返回给查看器的标头

将 CloudFront 配置为转发和缓存标头不会影响 CloudFront 返回给查看器的标头。CloudFront 返回从源获取的所有标头，只有少数几个例外。有关更多信息，请参阅相关主题：

- Amazon S3 源 – 请参阅CloudFront 删除或更新的 HTTP 响应标头。
- 自定义源 – 请参阅CloudFront 移除或替换的 HTTP 响应标头。

使用策略来控制缓存键

使用 CloudFront 缓存策略，您可以为在 CloudFront 边缘站点中缓存的对象，指定 CloudFront 将包含在缓存键中的 HTTP 标头、Cookie 和查询字符串。缓存键是缓存中每个对象的唯一标识符，它决定查看器的 HTTP 请求是否导致缓存命中。

如果查看器请求生成与先前请求相同的缓存键，并且该缓存键的对象位于边缘站点的缓存中且有效，则会发生缓存命中。当存在缓存命中时，该对象将从 CloudFront 边缘站点提供给查看器，这具有以下好处：

- 减少了源服务器上的负载
- 缩短了查看器的延迟

缓存键中包含的值越少，缓存命中结果的可能性越高。这可以为您的网站和应用程序带来更好的性能，因为可以得到更高的缓存命中率（导致缓存命中的查看器请求的比率更高）。有关更多信息，请参阅了解缓存键。

要控制缓存键，请使用 CloudFront 缓存策略。将缓存策略附加到 CloudFront 分配中的一个或多个缓存行为。

您还可以使用缓存策略为 CloudFront 缓存中的对象指定存活时间（TTL）设置，并允许 CloudFront 请求和缓存压缩对象。

主题

了解缓存策略

利用缓存策略，您可以通过控制缓存键中包含的值（URL 查询字符串、HTTP 标头和 Cookie）来提高缓存命中率。CloudFront 为常见使用案例提供了一些预定义的源请求策略（称为托管策略）。您可以使用这些托管策略，也可以创建特定于您的需求的缓存策略。有关托管策略的更多信息，请参阅使用托管式缓存策略。

缓存策略包含以下设置，这些设置的分类如下：策略信息、生存时间 (TTL) 设置和缓存键设置。

策略信息

名称

用于标识缓存策略的名称。在控制台中，可以使用名称将缓存策略附加到缓存行为。

说明

用于描述缓存策略的注释。虽然此选项是可选的，但它可以帮助您确定缓存策略的用途。

生存时间 (TTL) 设置

将生存时间（TTL）设置与 `Cache-Control` 和 `Expires` HTTP 标头（如果它们在源响应中）配合使用，可以确定 CloudFront 缓存中的对象保持有效的时间。

最小 TTL

您希望对象在 CloudFront 缓存中保留的最短时间（以秒为单位），在此时间之后，CloudFront 会向您的源转发另一个请求以确定此对象是否已更新。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

最大 TTL

在 CloudFront 向源发送另一个请求以查看对象是否已更新之前，对象在 CloudFront 缓存中保留的最大时间量（以秒为单位）。仅当源对象随对象发送 `Cache-Control` 或 `Expires` 标头时，CloudFront 才使用此设置。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

默认 TTL

您希望对象在 CloudFront 缓存中保留的默认时间（以秒为单位），在此时间之后，CloudFront 会向您的源转发另一个请求以确定此对象是否已更新。仅当源不随对象发送 `Cache-Control` 或 `Expires` 标头时，CloudFront 才使用此设置的值作为对象的 TTL。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

> **Note**
>
> 如果最小 TTL、最大 TTL 和默认 TTL 设置都设置为 0，则会禁用 CloudFront 缓存。

缓存键设置

缓存键设置指定 CloudFront 包含在缓存键中的查看器请求的值。这些值可以包括 URL 查询字符串、HTTP 标头和 Cookie。缓存键中包含的值将自动包含在 CloudFront 发送到源的请求（称为源请求）中。有关在不影响缓存键的情况下控制源请求的信息，请参阅使用策略来控制源请求。

缓存键设置包括：

- 标头
- Cookie
- 查询字符串
- 压缩支持

标头

CloudFront 包含在缓存键和源请求中的查看器请求中的 HTTP 标头。对于标头，您可以选择下列设置之一：

- 无 – 查看器请求中的 HTTP 标头不 会包含在缓存键中，也不 会自动包含在源请求中。
- Include the following headers（包含以下标头）– 您可以指定查看器请求中的哪些 HTTP 标头包含在缓存键中，并且会自动包含在源请求中。

在使用 Include the following headers（包含以下标头）设置时，可以按 HTTP 标头的名称（而不是值）指定它们。例如，请考虑以下 HTTP 标头：

```
Accept-Language: en-US,en;q=0.5
```

在此情况下，您可以将标头指定为 `Accept-Language`，而不是指定为 `Accept-Language: en-US,en;q=0.5`。但是，CloudFront 会将完整标头（包括其值）包含在缓存键和源请求中。

您还可以将 CloudFront 生成的某些标头包含在缓存键中。有关更多信息，请参阅 the section called “添加 CloudFront 请求标头”。

Cookie

CloudFront 包含在缓存键和源请求中的查看器请求中的 Cookie。对于 Cookie，您可以选择下列设置之一：

- 无 – 查看器请求中的 Cookie 不 会包含在缓存键中，也不 会自动包含在源请求中。

- 全部 – 查看器请求中的所有 Cookie 都包含在缓存键中，也会自动包含在源请求中。
- Include specified cookies（包含指定 Cookie）– 您可以指定查看器请求中的哪些 Cookie 包含在缓存键中，并且会自动包含在源请求中。
- Include all cookies except（包含全部 Cookie -例外项）– 您可以指定查看器请求中的哪些 Cookie 不 包含在缓存键中，并且不会自动包含在源请求中。所有其他 Cookie（您指定的 Cookie 除外）都 包含在缓存键中，并且会自动包含在源请求中。

在使用 Include specified cookies（包含指定 Cookie）或 Include all cookies except（包含全部 Cookie -例外项）设置时，可以按 Cookie 的名称（而不是值）指定它们。例如，请考虑以下 `Cookie` 标头：

```
Cookie: session_ID=abcd1234
```

在此情况下，您可以将 Cookie 指定为 `session_ID`，而不是指定为 `session_ID=abcd1234`。但是，CloudFront 会将完整 Cookie（包括其值）包含在缓存键和源请求中。

查询字符串

CloudFront 包含在缓存键和源请求中的查看器请求中的 URL 查询字符串。对于查询字符串，可以选择下列设置之一：

- 无 – 查看器请求中的查询字符串不 会包含在缓存键中，也不 会自动包含在源请求中。
- 全部 – 查看器请求中的所有查询字符串都包含在缓存键中，并且也会自动包含在源请求中。
- Include specified query strings（包含指定查询字符串）– 您可以指定查看器请求中的哪些查询字符串包含在缓存键中，并且会自动包含在源请求中。
- Include all query strings except（包含全部查询字符串-例外项）– 您可以指定查看器请求中的哪些查询字符串不 包含在缓存键中，并且不 会自动包含在源请求中。所有其他查询字符串（您指定的查询字符串除外）都 包含在缓存键中，并且会自动包含在源请求中。

在使用 Include specified query strings（包含指定查询字符串）或 Include all query strings except（包含全部查询字符串-例外项）设置时，可以按查询字符串的名称（而不是值）指定它们。例如，请考虑以下 URL 路径：

```
/content/stories/example-story.html?split-pages=false
```

在此情况下，您可以将查询字符串指定为 `split-pages`，而不是指定为 `split-pages=false`。但是，CloudFront 会将完整的查询字符串（包括其值）包含在缓存键和源请求中。

压缩支持

利用这些设置，CloudFront 可以在查看器支持 Gzip 或 Brotli 压缩格式时，请求和缓存以该压缩格式压缩的对象。这些设置还允许 CloudFront 压缩功能发挥作用。查看器通过 `Accept-Encoding` HTTP 标头表示它们支持这些压缩格式。

> **Note**
>
> Chrome 和 Firefox Web 浏览器仅在使用 HTTPS 发送请求时支持 Brotli 压缩。这些浏览器不支持带 HTTP 请求的 Brotli。

在满足以下任一条件时，启用这些设置：

- 当查看器支持 Gzip 压缩对象时，您的源将返回这些对象（请求包含带 `gzip` 的 `Accept-Encoding` HTTP 标头作为值）。在此情况下，使用启用了 Gzip 设置（在 CloudFront API、AWS 开发工具包、AWS CLI 或 AWS CloudFormation 中将 `EnableAcceptEncodingGzip` 设置为 `true`）。
- 当查看器支持 Brotli 压缩对象时，您的源将返回这些对象（请求包含带 `br` 的 `Accept-Encoding` HTTP 标头作为值）。在此情况下，使用启用了 Brotli 设置（在 CloudFront API、AWS 开发工具包、AWS CLI 或 AWS CloudFormation 中将 `EnableAcceptEncodingBrotli` 设置为 `true`）。
- 此缓存策略附加到的缓存行为将通过 CloudFront 压缩进行配置。在此情况下，可以为 Gzip 和/或 Brotli 启用缓存。启用 CloudFront 压缩后，为两种格式启用缓存可帮助降低将数据传输到 Internet 的成本。

> **Note**
>
> 如果为其中一种或两种压缩格式启用缓存，则不要在与相同缓存行为关联的源请求策略中包含 `Accept-Encoding` 标头。当为这些格式中的任一格式启用缓存时，CloudFront 始终在源请求中包含此标头，因此在源请求策略中包含 `Accept-Encoding` 不起作用。

如果源服务器未返回 Gzip 或 Brotli 压缩对象，或者缓存行为未通 CloudFront 压缩进行配置，则不要为压缩对象启用缓存。如果这样做的话，可能会导致缓存命中率下降。

下面说明了这些设置如何影响 CloudFront 分配。以下所有方案都假定查看器请求包含 `Accept-Encoding` 标头。如果查看器请求不包含 `Accept-Encoding` 标头，则 CloudFront 不会将此标头包含在缓存键和相应的源请求中。

在为两种压缩格式支持缓存压缩对象的情况下

如果查看器同时支持 Gzip 和 Brotli，即，如果 gzip 和 br 值都在查看器请求中的 Accept-Encoding 标头中，CloudFront 将执行以下操作：

- 将标头标准化为 Accept-Encoding: br,gzip 并将标准化的标头包含在缓存键中。缓存键不包含查看器发送的 Accept-Encoding 标头中的其他值。
- 如果边缘站点在缓存中具有与请求匹配的 Brotli 或 Gzip 压缩对象，并且未过期，则边缘站点会将此对象返回给查看器。
- 如果边缘站点在缓存中没有与请求匹配的 Brotli 或 Gzip 压缩对象，并且未过期，则 CloudFront 会将标准化的标头（Accept-Encoding: br,gzip）包含在相应的源请求中。源请求不包含查看器发送的 Accept-Encoding 标头中的其他值。

如果查看器支持一种压缩格式，而不支持另一种压缩格式例如，如果 gzip 是查看器请求中 Accept-Encoding 标头中的值，而 br 不是，则 CloudFront 将执行以下操作：

- 将标头标准化为 Accept-Encoding: gzip 并将标准化的标头包含在缓存键中。缓存键不包含查看器发送的 Accept-Encoding 标头中的其他值。
- 如果边缘站点在缓存中具有与请求匹配的 Gzip 压缩对象，并且未过期，则边缘站点会将此对象返回给查看器。
- 如果边缘站点在缓存中没有与请求匹配的 Gzip 压缩对象，并且未过期，则 CloudFront 会将标准化的标头（Accept-Encoding: gzip）包含在相应的源请求中。源请求不包含查看器发送的 Accept-Encoding 标头中的其他值。

要了解当查看器支持 Brotli 而非 Gzip 时，CloudFront 将执行的操作，请在前面示例中将两种压缩格式相互替换。

如果查看器不支持 Brotli 或 Gzip，即，查看器请求中的 Accept-Encoding 标头不包含 br 或 gzip 作为值，则 CloudFront：

- 不会将 Accept-Encoding 标头包含在缓存键中。
- 将 Accept-Encoding: identity 包含在相应的源请求中。源请求不包含查看器发送的 Accept-Encoding 标头中的其他值。

在为一种压缩格式（而非另一种压缩格式）支持缓存压缩对象的情况下

例如，如果查看器支持已启用缓存的格式，例如，为 Gzip 启用缓存压缩对象并且查看器支持 Gzip（gzip 是查看器请求中 Accept-Encoding 标头中的值之一），CloudFront 将执行以下操作：

- 将标头标准化为 Accept-Encoding: gzip 并将标准化的标头包含在缓存键中。

- 如果边缘站点在缓存中具有与请求匹配的 Gzip 压缩对象，并且未过期，则边缘站点会将此对象返回给查看器。
- 如果边缘站点在缓存中没有与请求匹配的 Gzip 压缩对象，并且未过期，则 CloudFront 会将标准化的标头（`Accept-Encoding: gzip`）包含在相应的源请求中。源请求不包含查看器发送的 `Accept-Encoding` 标头中的其他值。

当查看器同时支持 Gzip 和 Brotli（查看器请求中的 `Accept-Encoding` 标头包含 `gzip` 和 `br` 作为值）时，此行为是相同的，因为在这种情况下，不支持为 Brotli 缓存压缩对象。

要了解当为 Brotli 而非 Gzip 支持缓存压缩对象时，CloudFront 将执行的操作，请在前面示例中将两种压缩格式相互替换。

如果查看器不支持已启用缓存的压缩格式（查看器请求中的 `Accept-Encoding` 标头不包含该格式的值），则 CloudFront：

- 不会将 `Accept-Encoding` 标头包含在缓存键中。
- 将 `Accept-Encoding: identity` 包含在相应的源请求中。源请求不包含查看器发送的 `Accept-Encoding` 标头中的其他值。

在不支持为两种压缩格式缓存压缩对象的情况下

在不支持为两种压缩格式缓存压缩对象时，CloudFront 会像处理查看器请求中的任何其他 HTTP 标头一样处理 `Accept-Encoding` 标头。默认情况下，它不包括在缓存键中，也不包括在源请求中。与任何其他 HTTP 标头一样，可以将此标头包含在缓存策略或源请求策略中的标头列表中。

创建缓存策略

利用缓存策略，您可以通过控制缓存键中包含的值（URL 查询字符串、HTTP 标头和 Cookie）来提高缓存命中率。您可以使用 AWS Command Line Interface（AWS CLI）或 CloudFront API 在 CloudFront 控制台中创建缓存策略。

创建缓存策略后，将它附加到 CloudFront 分配中的一个或多个缓存行为。

Console

创建缓存策略（控制台）

1. 登录到 AWS Management Console 并通过以下网址在 CloudFront 控制台中打开 Policies (策略) 页面：https://console.aws.amazon.com/cloudfront/v4/home?#/policies。

2. 选择创建缓存策略。
3. 为此缓存策略选择所需的设置。有关更多信息，请参阅 了解缓存策略。
4. 完成后，选择 Create（创建）。

创建缓存策略后，可以将它附加到缓存行为。

将缓存策略附加到现有分配（控制台）

1. 在 CloudFront 控制台中打开 Distributions (分配) 页面，网址为 https://console.aws.amazon.com/cloudfront/v4/home#/distributions。
2. 选择要更新的分配，然后选择行为选项卡。
3. 选择要更新的缓存行为，然后选择编辑。

   或者，要创建新的缓存行为，请选择 Create behavior(创建行为)。
4. 在 Cache key and origin requests（缓存键和源请求）区域，请确保选择了 Cache policy and origin request policy（缓存策略和源请求策略）。
5. 对于 Cache policy（缓存策略），选择要附加到此缓存行为的缓存策略。
6. 在页面底部，选择 Save changes（保存更改）。

将缓存策略附加到新分配（控制台）

1. 通过 打开 CloudFront 控制台https://console.aws.amazon.com/cloudfront/v4/home
2. 选择 Create distribution（创建分配）。
3. 在 Cache key and origin requests（缓存键和源请求）区域，请确保选择了 Cache policy and origin request policy（缓存策略和源请求策略）。
4. 对于 Cache policy（缓存策略），选择要附加到此分发的原定设置缓存行为的缓存策略。
5. 为源、原定设置缓存行为和其他分配设置选择所需的设置。有关更多信息，请参阅 分配设置参考。
6. 完成后，选择 Create distribution（创建分配）。

CLI

要使用 AWS Command Line Interface (AWS CLI) 创建缓存策略，请使用 aws cloudfront create-cache-policy 命令。您可以使用输入文件来提供命令的输入参数，而不是将每个单独的参数指定为命令行输入。

创建缓存策略（带输入文件的 CLI）

1. 使用以下命令创建名为 cache-policy.yaml 的文件，其中包含 create-cache-policy 命令的所有输入参数。

```
aws cloudfront create-cache-policy --generate-cli-skeleton yaml-input > cache-policy.yaml
```

2. 打开刚创建的名为 cache-policy.yaml 的文件。编辑该文件以指定所需的缓存策略设置，然后保存该文件。您可以从该文件中删除可选字段，但不要删除必填字段。

   有关缓存策略设置的更多信息，请参阅了解缓存策略。

3. 使用以下命令通过 cache-policy.yaml 文件中的输入参数创建缓存策略。

```
aws cloudfront create-cache-policy --cli-input-yaml file://cache-policy.yaml
```

   记下命令输出中的 Id 值。这是缓存策略 ID，您需要它才能将缓存策略附加到 CloudFront 分配的缓存行为。

将缓存策略附加到现有分配（带输入文件的 CLI）

1. 使用以下命令保存要更新的 CloudFront 分配的分配配置。将 *distribution_ID* 替换为分配的 ID。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
 dist-config.yaml
```

2. 打开刚创建的名为 dist-config.yaml 的文件。编辑该文件，对要更新的每个缓存行为进行以下更改以使用缓存策略。

   - 在缓存行为中，添加名为 CachePolicyId 的字段。对于字段的值，请使用创建策略后记下的缓存策略 ID。
   - 从缓存行为中删除 MinTTL、MaxTTL、DefaultTTL 和 ForwardedValues 字段。这些设置是在缓存策略中指定的，因此不能将这些字段和一个缓存策略包含在相同的缓存行为中。
   - 将 ETag 字段重命名为 IfMatch，但不更改字段的值。

完成后保存该文件。

3. 使用以下命令更新分配以使用缓存策略。将 *distribution_ID* 替换为分配的 ID。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

将缓存策略附加到新分配（带输入文件的 CLI）

1. 使用以下命令创建名为 `distribution.yaml` 的文件，其中包含 create-distribution 命令的所有输入参数。

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. 打开刚创建的名为 `distribution.yaml` 的文件。在默认缓存行为中，在 `CachePolicyId` 字段中输入创建策略后记下的缓存策略 ID。继续编辑该文件以指定所需的分配设置，然后在完成后保存该文件。

   有关分配设置的更多信息，请参阅分配设置参考。

3. 使用以下命令通过 `distribution.yaml` 文件中的输入参数创建分配。

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

要使用 CloudFront API 创建缓存策略，请使用 CreateCachePolicy。有关您在此 API 调用中指定的字段的更多信息，请参阅了解缓存策略以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

创建缓存策略后，可以使用下列 API 调用之一将该缓存策略附加到缓存行为：

- 要将该配置附加到现有分配中的缓存行为，请使用 UpdateDistribution。
- 要将该配置附加到新分配中的缓存行为，请使用 CreateDistribution。

对于这两个 API 调用，请在缓存行为中的 `CachePolicyId` 字段中提供缓存策略的 ID。有关您在这些 API 调用中指定的其他字段的更多信息，请参阅 分配设置参考 以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

使用托管式缓存策略

CloudFront 提供了一组托管缓存策略，您可以将这些策略附加到分配的任意缓存行为。有了托管缓存策略，您无需编写或维护自己的缓存策略。托管策略使用已针对特定使用案例优化的设置。

要使用托管缓存策略，请将它附加到分配中的缓存行为。此过程与创建缓存策略时的过程相同，只不过您只需附加一个托管缓存策略，而不是创建新的缓存策略。您可以按名称（使用控制台）或 ID（使用 AWS CLI 或开发工具包）附加策略。以下部分列出了名称和 ID。

有关更多信息，请参阅 创建缓存策略。

下面的主题介绍了您可以使用的托管缓存策略。

主题

- Amplify
- CachingDisabled
- CachingOptimized
- CachingOptimizedForUncompressedObjects
- Elemental-MediaPackage
- UseOriginCacheControlHeaders
- UseOriginCacheControlHeaders-QueryStrings

Amplify

在 CloudFront 控制台中查看此策略

此策略适合用于作为 AWS Amplify Web 应用程序的源。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`2e54312d-136d-493c-8eb9-b001f22f67d2`

此策略包含以下设置：

- 最小 TTL：2 秒
- 最大 TTL：600 秒（10 分钟）
- 原定设置 TTL：2 秒
- 缓存键中包含的标头：
  - `Authorization`
  - `CloudFront-Viewer-Country`
  - `Host`

  还包含标准化的 `Accept-Encoding` 标头，因为已启用缓存压缩对象设置。有关更多信息，请参阅压缩支持。
- 缓存键中包含的 Cookie：包含所有 Cookie。
- 缓存键中包含的查询字符串：包含所有查询字符串。
- 缓存压缩对象设置：已启用。有关更多信息，请参阅压缩支持。

CachingDisabled

在 CloudFront 控制台中查看此策略

此策略禁用缓存。此策略对于动态内容和不可缓存的请求很有用。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`4135ea2d-6df8-44a3-9df3-4b5a84be39ad`

此策略包含以下设置：

- 最小 TTL：0 秒
- 最大 TTL：0 秒
- 原定设置 TTL：0 秒
- 缓存键中包含的标头：无
- 缓存键中包含的 Cookie：无
- 缓存键中包含的查询字符串：无
- 缓存压缩对象设置：已禁用

CachingOptimized

在 CloudFront 控制台中查看此策略

此策略旨在通过最大程度地减少 CloudFront 在缓存键中包含的值来提高缓存效率。CloudFront 不在缓存键中包含任何查询字符串或 Cookie，并且只包含标准化 `Accept-Encoding` 标头。这使 CloudFront 能够在源返回对象或在启用 CloudFront 边缘压缩时以 Gzip 和 Brotli 压缩格式分别缓存对象。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`658327ea-f89d-4fab-a63d-7e88639e58f6`

此策略包含以下设置：

- 最小 TTL：1 秒。
- 最大 TTL：31536000 秒（365 天）。
- 原定设置 TTL：86400 秒（24 小时）。
- 缓存键中包含的标头：未明确包含任何标头。包含标准化的 `Accept-Encoding` 标头，因为已启用缓存压缩对象设置。有关更多信息，请参阅压缩支持。
- 缓存键中包含的 Cookie：无。
- 缓存键中包含的查询字符串：无。
- 缓存压缩对象设置：已启用。有关更多信息，请参阅压缩支持。

CachingOptimizedForUncompressedObjects

在 CloudFront 控制台中查看此策略

此策略旨在通过最大程度地减少缓存键中包含的值来提高缓存效率。不包括查询字符串、标头或 Cookie。此策略与前一个策略相同，但它禁用了缓存压缩对象设置。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`b2884449-e4de-46a7-ac36-70bc7f1ddd6d`

此策略包含以下设置：

- 最小 TTL：1 秒

- 最大 TTL：31536000 秒（365 天）
- 原定设置 TTL：86400 秒（24 小时）
- 缓存键中包含的标头：无
- 缓存键中包含的 Cookie：无
- 缓存键中包含的查询字符串：无
- 缓存压缩对象设置：已禁用

Elemental-MediaPackage

在 CloudFront 控制台中查看此策略

此策略旨在用于作为 AWS Elemental MediaPackage 终端节点的源。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`08627262-05a9-4f76-9ded-b50ca2e3a84f`

此策略包含以下设置：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（365 天）
- 原定设置 TTL：86400 秒（24 小时）
- 缓存键中包含的标头：
  - `Origin`

  还包含标准化的 `Accept-Encoding` 标头，因为已经为 Gzip 启用了缓存压缩对象设置。有关更多信息，请参阅压缩支持。
- 缓存键中包含的 Cookie：无
- 缓存键中包含的查询字符串：
  - `aws.manifestfilter`
  - `start`
  - `end`
  - `m`
- 缓存压缩对象设置：已对 Gzip 启用。有关更多信息，请参阅压缩支持。

UseOriginCacheControlHeaders

在 CloudFront 控制台中查看此策略

此策略设计用于符合以下条件的源：返回 `Cache-Control` HTTP 响应标头，且不会根据查询字符串中存在的值提供不同内容。如果源会根据查询字符串中存在的值提供不同内容，请考虑使用 UseOriginCacheControlHeaders-QueryStrings。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`83da9c7e-98b4-4e11-a168-04f0df8e2c65`

此策略包含以下设置：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（365 天）
- 原定设置 TTL：0 秒
- 缓存键中包含的标头：
  - `Host`
  - `Origin`
  - `X-HTTP-Method-Override`
  - `X-HTTP-Method`
  - `X-Method-Override`

  还包含标准化的 `Accept-Encoding` 标头，因为已启用缓存压缩对象设置。有关更多信息，请参阅压缩支持。
- 缓存键中包含的 Cookie：包含所有 Cookie。
- 缓存键中包含的查询字符串：无。
- 缓存压缩对象设置：已启用。有关更多信息，请参阅压缩支持。

UseOriginCacheControlHeaders-QueryStrings

在 CloudFront 控制台中查看此策略

此策略设计用于符合以下条件的源：返回 `Cache-Control` HTTP 响应标头，且会根据查询字符串中存在的值提供不同内容。如果源不会根据查询字符串中存在的值提供不同内容，请考虑使用 UseOriginCacheControlHeaders。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`4cc15a8a-d715-48a4-82b8-cc0b614638fe`

此策略包含以下设置：

- 最小 TTL：0 秒
- 最大 TTL：31536000 秒（365 天）
- 原定设置 TTL：0 秒
- 缓存键中包含的标头：
  - `Host`
  - `Origin`
  - `X-HTTP-Method-Override`
  - `X-HTTP-Method`
  - `X-Method-Override`

  还包含标准化的 `Accept-Encoding` 标头，因为已启用缓存压缩对象设置。有关更多信息，请参阅压缩支持。
- 缓存键中包含的 Cookie：包含所有 Cookie。
- 缓存键中包含的查询字符串：包含所有查询字符串。
- 缓存压缩对象设置：已启用。有关更多信息，请参阅压缩支持。

了解缓存键

缓存键 确定发送到 CloudFront 边缘站点的查看器请求是否导致缓存命中。缓存键是缓存中对象的唯一标识符。缓存中的每个对象都有一个唯一的缓存键。

如果查看器请求生成与先前请求相同的缓存键，并且该缓存键的对象位于边缘站点的缓存中且有效，则会发生缓存命中。当存在缓存命中时，请求的对象将从 CloudFront 边缘站点提供给查看器，这具有以下好处：

- 减少了源服务器上的负载
- 缩短了查看器的延迟

缓存命中率越高（当查看器请求比例较高时，会发生缓存命中），网站或应用程序的性能就越佳。提高缓存命中率的一种方法是仅在缓存键中包含最少的必要值。有关更多信息，请参阅以下部分。

您可以使用缓存策略修改缓存键中的值（URL 查询字符串、HTTP 标头和 Cookie）。（也可以使用 Lambda@Edge 函数修改缓存键。）在修改缓存键之前，请务必了解应用程序的设计方式以及它何时以及如何根据查看器请求的特征提供不同的响应。当查看器请求中的某个值确定源返回的响应时，您应在缓存键中包含该值。但是，如果您在缓存键中包含的值不会影响源返回的响应，则可能最终会缓存重复的对象。

默认缓存键

默认情况下，CloudFront 分配的缓存键包括以下信息：

- CloudFront 分配的域名（例如，d111111abcdef8.cloudfront.net）
- 请求的对象的 URL 路径（例如 `/content/stories/example-story.html`）

> **Note**
>
> OPTIONS 方法包含在 OPTIONS 请求的缓存键中。这意味着 OPTIONS 请求的响应将与 GET 和 HEAD 请求的响应单独缓存。

默认情况下，查看器请求中的其他值不会包含在缓存键中。请考虑来自 Web 浏览器的以下 HTTP 请求。

```
            GET /content/stories/example-story.html?ref=0123abc&split-pages=false
HTTP/1.1
            Host: d111111abcdef8.cloudfront.net
            User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/68.0
            Accept: text/html,*/*
            Accept-Language: en-US,en
            Cookie: session_id=01234abcd
            Referer: https://news.example.com/
```

当与此示例类似的查看器请求到达 CloudFront 边缘站点时，CloudFront 将使用缓存键来确定是否存在缓存命中。默认情况下，缓存键中仅包含请求的以下组件：`/content/stories/example-story.html` 和 `d111111abcdef8.cloudfront.net`。如果请求的对象不在缓存中（缓存未命中），则 CloudFront 会向源发送请求以获取该对象。获取该对象后，CloudFront 会将它返回到查看器并将它存储在边缘站点的缓存中。

当 CloudFront 收到由缓存键确定的同一对象的另一个请求时，CloudFront 会立即将缓存对象提供给查看器，而不会向源发送请求。例如，请考虑以下 HTTP 请求，该请求在上一个请求之后到达。

```
            GET /content/stories/example-story.html?ref=xyz987&split-pages=true
HTTP/1.1
            Host: d111111abcdef8.cloudfront.net
            User-Agent: Mozilla/5.0 AppleWebKit/537.36 Chrome/83.0.4103.116
            Accept: text/html,*/*
            Accept-Language: en-US,en
            Cookie: session_id=wxyz9876
            Referer: https://rss.news.example.net/
```

虽然该请求与上一个请求都对应于相同的对象，但它与上一个请求不同。它具有不同的 URL 查询字符串、User-Agent 和 Referer 标头，以及不同的 session_id Cookie。但默认情况下，所有这些值都不是缓存键的一部分，因此第二个请求会导致发生缓存命中。

自定义缓存键

在某些情况下，即使这样做可能会导致更少的缓存命中，您仍可能希望在缓存键中包含更多信息。您可以使用缓存策略指定要包含在缓存键中的内容。

例如，如果您的源服务器使用查看器请求中的 Accept-Language HTTP 标头来根据查看器的语言返回不同的内容，则您可能希望在缓存键中包含此标头。在执行该操作时，CloudFront 会使用此标头确定缓存命中，并在源请求（当缓存未命中时，CloudFront 发送到源的请求）中包含标头。

在缓存键中包含其他值的一个潜在后果是，由于查看器请求中可能发生的变化，CloudFront 最终可能会缓存重复的对象。例如，查看器可能会为 Accept-Language 标头发送以下任意值：

- en-US,en
- en,en-US
- en-US, en
- en-US

所有这些不同的值都表明查看器的语言是英语，但变体可能会导致 CloudFront 将同一个对象缓存多次。这会减少缓存命中数并增加源请求的数量。您可以避免此重复情况，方法是不在缓存键中包含 Accept-Language 标头，而是将您的网站或应用程序配置为将不同的 URL 用于不同语言的内容（例如 /en-US/content/stories/example-story.html）。

对于要包含在缓存键中的任何给定值，您应确保了解查看器请求中可能会出现的不同变体的数量。对于某些请求值，将它们包含在缓存键中几乎没有意义。例如，User-Agent 标头可以有数千个唯一变体，因此通常不适合将它作为包含在缓存键中的候选项。也不适合将具有用户特定的值或会话特定的值并且在数千个（甚至数百万个）请求中唯一的 Cookie 作为包含在缓存键中的候选项。如果您确实在缓存键中包含这些值，则每个唯一变体都会导致缓存中存在对象的另一个副本。如果对象的这些副本都不是唯一的，或者如果您最终获得了大量略微不同的对象，而每个对象只获得少量的缓存命中，则可能需要考虑另一种方法。您可以从缓存键中排除这些高度可变的值，也可以将对象标记为不可缓存。

在自定义缓存键时，请保持谨慎。有时这样做是可取的，但可能会产生意想不到的后果，例如缓存重复的对象、降低缓存命中率以及增加源请求的数量。如果您的源网站或应用程序需要接收来自查看器请求的特定值以进行分析、遥测或实现其他目的，但这些值不会更改源返回的对象，请使用源请求策略在源请求中包含这些值，但不 将它们包含在缓存键中。

使用策略来控制源请求

当发送到 CloudFront 的查看器请求导致缓存未命中（请求的对象未在边缘站点缓存）时，CloudFront 会向源发送请求以检索对象。这称为源请求。源请求始终包含来自查看器请求的以下信息：

- URL 路径（仅路径，不包含 URL 查询字符串或域名）
- 请求正文（如果有）
- CloudFront 在每个源请求中自动包含的 HTTP 标头，包括 `Host`、`User-Agent` 和 `X-Amz-Cf-Id`。

默认情况下，查看器请求中的其他信息（如 URL 查询字符串、HTTP 标头和 Cookie）不包含在源请求中。（例外：使用旧缓存设置时，CloudFront 默认将标头转发到您的源。）然而，您可能希望在源处接收其它一些此类信息，例如收集数据以进行分析或遥测。您可以使用源请求策略 控制源请求中包含的信息。

源请求策略与控制缓存键的缓存策略是分开的。通过这种方法，您可以在源端接收其他信息，并保持良好的缓存命中率（查看器请求导致缓存命中的比率）。您可以通过单独控制哪些信息包含在源请求中（使用源请求策略）以及哪些信息包含在缓存键中（使用缓存策略）来做到这一点。

虽然这两种策略是分开的，但它们却相关联。您在缓存键中包含的所有 URL 查询字符串、HTTP 标头和 Cookie（使用缓存策略）都将自动包含在源请求中。使用源请求策略指定要包含在源请求中但不 包含在缓存键中的信息。与缓存策略一样，您可以将源请求策略附加到 CloudFront 分配中的一个或多个缓存行为。

还可以使用源请求策略将其他 HTTP 标头添加到查看器请求中未包含的源请求。这些附加标头是 CloudFront 在发送源请求之前添加的，而标头值是根据查看器请求自动确定的。有关更多信息，请参阅 the section called “添加 CloudFront 请求标头”。

主题

了解源请求策略

CloudFront 为常见使用案例提供了一些预定义的源请求策略（称为托管策略）。您可以使用这些托管策略，也可以创建特定于您的需求的源请求策略。有关托管策略的更多信息，请参阅使用托管式源请求策略。

源请求策略包含以下设置，这些设置的分类如下：策略信息 和源请求设置。

策略信息

名称

用于标识源请求策略的名称。在控制台中，可以使用名称将源请求策略附加到缓存行为。

描述

描述源请求策略的注释。该项为可选项。

源请求设置

源请求设置指定查看器请求中包含的值，这些值包含在 CloudFront 发送到源的请求（称为源请求）中。这些值可以包括 URL 查询字符串、HTTP 标头和 Cookie。您指定的值包含在源请求中，但不会包含在缓存键中。有关控制缓存键的信息，请参阅使用策略来控制缓存键。

标头

CloudFront 包含在源请求中的查看器请求中的 HTTP 标头。对于标头，您可以选择下列设置之一：

- 无 – 查看器请求中的 HTTP 标头不 会包含在源请求中。
- 所有查看器标头 – 查看器请求中的所有 HTTP 标头都包含在源请求中。
- 所有查看器标头以及下列 CloudFront 标头 – 查看器请求中的所有 HTTP 标头都包含在源请求中。此外，您可以指定要添加到源请求中的 CloudFront 标头。有关 CloudFront 标头的更多信息，请参阅the section called “添加 CloudFront 请求标头”。
- Include the following headers（包含以下标头）– 您可以指定哪些 HTTP 标头包含在源请求中。

> **Note**
>
> 请勿指定已包含在源自定义标头设置中的标头。有关更多信息，请参阅 配置 CloudFront 以便向源请求添加自定义标头。

- 除以下范围之外的所有查看器标头–您指定哪些 HTTP 标头不包含在源请求中。除了指定的标头外，查看器请求中的所有其他 HTTP 标头都包括在内。

当您使用所有查看器标头以及下列 CloudFront 标头、包括以下标头或除以下范围之外的所有查看器标头设置时，可以仅按 HTTP 标头的名称指定它们。CloudFront 会将完整标头（包括其值）包含在源请求中。

> **Note**
>
> 当您使用除以下范围之外的所有查看器标头设置来删除查看器的 `Host` 标头时，CloudFront 会在源请求中添加一个包含源域名的新 `Host` 标头。

Cookie

CloudFront 包含在源请求中的查看器请求中的 Cookie。对于 Cookie，您可以选择下列设置之一：

- 无 – 查看器请求中的 Cookie 不 会包含在源请求中。
- 全部 – 查看器请求中的所有 Cookie 都包含在源请求中。
- 包括以下 Cookie – 您可以指定查看器请求中的哪些 Cookie 包含在源请求中。
- 除以下范围之外的所有 Cookie – 您可以指定查看器请求中的哪些 Cookie 不包含在源请求中。查看器请求中的所有其他 Cookie 都包含在内。

在使用包括以下 Cookie 或 除以下范围之外的所有 Cookie 设置时，可以仅按 Cookie 的名称指定它们。CloudFront 会将完整 Cookie（包括其值）包含在源请求中。

查询字符串

CloudFront 包含在源请求中的查看器请求中的 URL 查询字符串。对于查询字符串，可以选择下列设置之一：

- 无 – 查看器请求中的查询字符串不 会包含在源请求中。
- 全部 – 查看器请求中的所有查询字符串都包含在源请求中。
- 包含指定以下字符串 – 您可以指定查看器请求中的哪些查询字符串包含在源请求中。
- 除以下范围之外的所有查询字符串 – 您可以指定查看器请求中的哪些查询字符串不包含在源请求中。所有其他查询字符串都包括在内。

在使用包括以下查询字符串或除以下范围之外的所有查询字符串设置时，您可以仅按查询字符串的名称指定它们。CloudFront 会将完整的查询字符串（包括其值），包含在源请求中。

创建源请求策略

您可以使用源请求策略控制包含在 CloudFront 发送到源的请求中的值（URL 查询字符串、HTTP 标头和 Cookie）。您可以使用 AWS Command Line Interface（AWS CLI）或 CloudFront API 在 CloudFront 控制台中创建源请求策略。

创建源请求策略后，可以将它附加到 CloudFront 分配中的一个或多个缓存行为。

源请求策略不是必需的。如果缓存行为未附加源请求策略，则源请求只会包含缓存策略中指定的所有值。

> **Note**
>
> 要使用源请求策略，缓存行为还必须使用缓存策略。如果没有缓存策略，则无法在缓存行为中使用源请求策略。

Console

创建源请求策略（控制台）

1. 登录到 AWS Management Console 并通过以下网址在 CloudFront 控制台中打开 Policies (策略) 页面：https://console.aws.amazon.com/cloudfront/v4/home?#/policies。
2. 选择 Origin request（源请求），然后选择 Create origin request policy（创建源请求策略）。
3. 为此源请求策略选择所需的设置。有关更多信息，请参阅 了解源请求策略。
4. 完成后，选择 Create（创建）。

创建源请求策略后，可以将它附加到缓存行为。

将源请求策略附加到现有分配（控制台）

1. 在 CloudFront 控制台中打开 Distributions (分配) 页面，网址为 https://console.aws.amazon.com/cloudfront/v4/home#/distributions。
2. 选择要更新的分配，然后选择行为选项卡。
3. 选择要更新的缓存行为，然后选择编辑。

   或者，要创建新的缓存行为，请选择 Create behavior(创建行为)。

4. 在 Cache key and origin requests（缓存键和源请求）区域，请确保选择了 Cache policy and origin request policy（缓存策略和源请求策略）。
5. 对于 Origin request policy（源请求策略），选择要附加到此缓存行为的源请求策略。
6. 在页面底部，选择 Save changes（保存更改）。

将源请求策略附加到新分配（控制台）

1. 通过 打开 CloudFront 控制台https://console.aws.amazon.com/cloudfront/v4/home
2. 选择 Create distribution（创建分配）。
3. 在 Cache key and origin requests（缓存键和源请求）区域，请确保选择了 Cache policy and origin request policy（缓存策略和源请求策略）。
4. 对于 Origin request policy（源请求策略），选择要附加到此分发的原定设置缓存行为的源请求策略。
5. 为源、原定设置缓存行为和其他分配设置选择所需的设置。有关更多信息，请参阅 分配设置参考。
6. 完成后，选择 Create distribution（创建分配）。

CLI

要使用 AWS Command Line Interface (AWS CLI) 创建源请求策略，请使用 aws cloudfront create-origin-request-policy 命令。您可以使用输入文件来提供命令的输入参数，而不是将每个单独的参数指定为命令行输入。

创建源请求策略（带输入文件的 CLI）

1. 使用以下命令创建名为 `origin-request-policy.yaml` 的文件，其中包含 create-origin-request-policy 命令的所有输入参数。

```
aws cloudfront create-origin-request-policy --generate-cli-skeleton yaml-input >
 origin-request-policy.yaml
```

2. 打开刚创建的名为 `origin-request-policy.yaml` 的文件。编辑该文件以指定所需的源请求策略设置，然后保存该文件。您可以从该文件中删除可选字段，但不要删除必填字段。

   有关源请求策略设置的更多信息，请参阅了解源请求策略。
3. 使用以下命令通过 `origin-request-policy.yaml` 文件中的输入参数创建源请求策略。

```
aws cloudfront create-origin-request-policy --cli-input-yaml file://origin-
request-policy.yaml
```

记下命令输出中的 `Id` 值。这是源请求策略 ID，您需要它才能将源请求策略附加到 CloudFront 分发的缓存行为。

将源请求策略附加到现有分配（带输入文件的 CLI）

1. 使用以下命令保存要更新的 CloudFront 分配的分配配置。将 *distribution_ID* 替换为分配的 ID。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
 dist-config.yaml
```

2. 打开刚创建的名为 `dist-config.yaml` 的文件。编辑该文件，对要更新的每个缓存行为进行以下更改以使用源请求策略。

   - 在缓存行为中，添加名为 `OriginRequestPolicyId` 的字段。对于字段的值，请使用创建策略后记下的源请求策略 ID。
   - 将 `ETag` 字段重命名为 `IfMatch`，但不更改字段的值。

   完成后保存该文件。

3. 使用以下命令更新分配以使用源请求策略。将 *distribution_ID* 替换为分配的 ID。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

将源请求策略附加到新分配（带输入文件的 CLI）

1. 使用以下命令创建名为 `distribution.yaml` 的文件，其中包含 create-distribution 命令的所有输入参数。

```
```

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. 打开刚创建的名为 `distribution.yaml` 的文件。在默认缓存行为中，在 `OriginRequestPolicyId` 字段中输入创建策略后记下的源请求策略 ID。继续编辑该文件以指定所需的分配设置，然后在完成后保存该文件。

   有关分配设置的更多信息，请参阅分配设置参考。

3. 使用以下命令通过 `distribution.yaml` 文件中的输入参数创建分配。

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

要使用 CloudFront API 创建源请求策略，请使用 CreateOriginRequestPolicy。有关您在此 API 调用中指定的字段的更多信息，请参阅了解源请求策略以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

创建源请求策略后，可以使用下列 API 调用之一将该策略附加到缓存行为：

- 要将该配置附加到现有分配中的缓存行为，请使用 UpdateDistribution。
- 要将该配置附加到新分配中的缓存行为，请使用 CreateDistribution。

对于这两个 API 调用，请在缓存行为中的 `OriginRequestPolicyId` 字段中提供源请求策略的 ID。有关您在这些 API 调用中指定的其他字段的更多信息，请参阅 分配设置参考 以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

使用托管式源请求策略

CloudFront 提供了一组托管源请求策略，您可以将这些策略附加到分配的任意缓存行为。有了托管源请求策略，您无需编写或维护自己的源请求策略。托管策略使用已针对特定使用案例优化的设置。

要使用托管源请求策略，请将它附加到分配中的缓存行为。此过程与创建源请求策略时的过程相同，只不过您只需附加一个托管源请求策略，而不是创建新的源请求策略。您可以按名称（使用控制台）或 ID（使用 AWS CLI 或开发工具包）附加策略。以下部分列出了名称和 ID。

有关更多信息，请参阅 创建源请求策略。

下面的主题介绍了您可以使用的托管源请求策略。

主题

- AllViewer
- AllViewerAndCloudFrontHeaders-2022-06
- AllViewerExceptHostHeader
- CORS-CustomOrigin
- CORS-S3Origin
- Elemental-MediaTailor-PersonalizedManifests
- UserAgentRefererHeaders

AllViewer

在 CloudFront 控制台中查看此策略

此策略包含来自查看器请求的所有值（标头、Cookie 和查询字符串）。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

```
216adef6-5c7f-47e4-b989-5492eafa07d3
```

此策略包含以下设置：

- 源请求中包含的标头：查看器请求中的所有标头
- 源请求中包含的 Cookie：全部
- 源请求中包含的查询字符串：全部

AllViewerAndCloudFrontHeaders-2022-06

在 CloudFront 控制台中查看此策略

此策略包含来自查看器请求的所有值（标头、Cookie 和查询字符串），以及在 2022 年 6 月之前发布的所有 CloudFront 标头（不包含 2022 年 6 月之后发布的 CloudFront 标头）。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

33f36d7e-f396-46d9-90e0-52428a34d9dc

此策略包含以下设置：

- 源请求中包含的标头：查看器请求中的所有标头，以及以下 CloudFront 标头：
  - CloudFront-Forwarded-Proto
  - CloudFront-Is-Android-Viewer
  - CloudFront-Is-Desktop-Viewer
  - CloudFront-Is-IOS-Viewer
  - CloudFront-Is-Mobile-Viewer
  - CloudFront-Is-SmartTV-Viewer
  - CloudFront-Is-Tablet-Viewer
  - CloudFront-Viewer-Address
  - CloudFront-Viewer-ASN
  - CloudFront-Viewer-City
  - CloudFront-Viewer-Country
  - CloudFront-Viewer-Country-Name
  - CloudFront-Viewer-Country-Region
  - CloudFront-Viewer-Country-Region-Name
  - CloudFront-Viewer-Http-Version
  - CloudFront-Viewer-Latitude
  - CloudFront-Viewer-Longitude
  - CloudFront-Viewer-Metro-Code
  - CloudFront-Viewer-Postal-Code
  - CloudFront-Viewer-Time-Zone
  - CloudFront-Viewer-TLS
- 源请求中包含的 Cookie：全部
- 源请求中包含的查询字符串：全部

AllViewerExceptHostHeader

在 CloudFront 控制台中查看此策略

此策略不包括来自查看器请求的 Host 标头，但包括来自查看器请求的所有其他值（标头、Cookie 和查询字符串）。

此策略还包括 HTTP 协议、HTTP 版本、TLS 版本以及所有设备类型和查看器位置标头的其他 CloudFront 请求标头。

本策略适用于Amazon API Gateway 和 AWS Lambda 函数 URL 源。这些源要求 Host 标头包含源域名，而不是 CloudFront 分配的域名。将来自查看器请求的 Host 标头转发到这些源可能会使它们无法正常运行。

> **Note**
>
> 当您使用此托管源请求策略删除查看器的 Host 标头时，CloudFront 会在源请求中添加一个包含源域名的新 Host 标头。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`b689b0a8-53d0-40ab-baf2-68738e2966ac`

此策略包含以下设置：

- 源请求中包含的标头：查看器请求中除了 Host 标头之外的所有标头
- 源请求中包含的 Cookie：全部
- 源请求中包含的查询字符串：全部

CORS-CustomOrigin

在 CloudFront 控制台中查看此策略

此策略包含在源为自定义源时启用跨源资源共享 (CORS) 请求的标头。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`59781a5b-3903-41f3-afcb-af62929ccde1`

此策略包含以下设置：

- 源请求中包含的标头：

  - `Origin`
- 源请求中包含的 Cookie：无
- 源请求中包含的查询字符串：无

CORS-S3Origin

在 CloudFront 控制台中查看此策略

此策略包含在源为 Amazon S3 存储桶时启用跨源资源共享 (CORS) 请求的标头。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`88a5eaf4-2fd4-4709-b370-b4c650ea3fcf`

此策略包含以下设置：

- 源请求中包含的标头：
  - `Origin`
  - `Access-Control-Request-Headers`
  - `Access-Control-Request-Method`
- 源请求中包含的 Cookie：无
- 源请求中包含的查询字符串：无

Elemental-MediaTailor-PersonalizedManifests

在 CloudFront 控制台中查看此策略

此策略旨在与作为 AWS Elemental MediaTailor 端点的源结合使用。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`775133bc-15f2-49f9-abea-afb2e0bf67d2`

此策略包含以下设置：

- 源请求中包含的标头：
  - `Origin`

  - `Access-Control-Request-Headers`
  - `Access-Control-Request-Method`
  - `User-Agent`
  - `X-Forwarded-For`
- 源请求中包含的 Cookie：无
- 源请求中包含的查询字符串：全部

UserAgentRefererHeaders

在 CloudFront 控制台中查看此策略

此策略仅包含 `User-Agent` 和 `Referer` 标头。它不包含任何查询字符串或 Cookie。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`acba4595-bd28-49b8-b9fe-13317c0390fa`

此策略包含以下设置：

- 源请求中包含的标头：
  - `User-Agent`
  - `Referer`
- 源请求中包含的 Cookie：无
- 源请求中包含的查询字符串：无

添加 CloudFront 请求标头

您可以对 CloudFront 进行配置，以便向 CloudFront 从查看器收到的请求中添加特定 HTTP 标头并转发到源或边缘函数。这些 HTTP 标头的值基于查看器或查看器请求的特征。这些标头提供了有关查看器的设备类型、IP 地址、地理位置、请求协议（HTTP 或 HTTPS）、HTTP 版本、TLS 连接详情和 JA3 指纹的信息。

有了这些标头，您的源或 Edge 函数可以接收有关查看器的信息，而无需编写自己的代码来确定此信息。如果源根据这些标头中的信息返回不同的响应，您可以将它们包含在缓存键 中，以便 CloudFront 分别缓存这些响应。例如，源可能会根据查看器所在的国家/地区以特定语言回复内容，或回复根据特

定设备类型量身定制的内容。源也可能会将这些标头写入日志文件，您可以使用这些标头来确定有关查看器所在位置、它们使用的设备类型等信息。

要将这些标头包含在缓存键中，请使用缓存策略。有关更多信息，请参阅使用策略来控制缓存键和the section called “了解缓存键”。

要在源接收这些标头但不将其包含在缓存键中，请使用源请求策略。有关更多信息，请参阅 使用策略来控制源请求。

主题

用于确定查看器的设备类型的标头

您可以添加以下标头来确定查看器的设备类型。根据 `User-Agent` 标头的值，CloudFront 将这些标头的值设置为 `true` 或 `false`。如果某个设备归入多个类别中，则可能有多个值为 `true`。例如，对于一些平板电脑设备，CloudFront 会将 `CloudFront-Is-Mobile-Viewer` 和 `CloudFront-Is-Tablet-Viewer` 都设置为 `true`。

- `CloudFront-Is-Android-Viewer` – 当 CloudFront 确定查看器是采用 Android 操作系统的设备时，设置为 `true`。
- `CloudFront-Is-Desktop-Viewer` – 当 CloudFront 确定查看器是桌面设备时，设置为 `true`。
- `CloudFront-Is-IOS-Viewer` – 当 CloudFront 确定查看器是带 Apple 操作系统（如 iPhone、iPod touch 和一些 iPad devices.的设备）时，设置为 `true`。
- `CloudFront-Is-Mobile-Viewer` – 当 CloudFront 确定查看器是移动设备时，设置为 `true`。
- `CloudFront-Is-SmartTV-Viewer` – 当 CloudFront 确定查看器是智能电视时，设置为 `true`。
- `CloudFront-Is-Tablet-Viewer` – 当 CloudFront 确定查看器是平板电脑时，设置为 `true`。

用于确定查看器的位置的标头

您可以添加以下标头来确定查看器的位置。CloudFront 根据查看器的 IP 地址确定这些标头的值。对于这些标头值中的非 ASCII 字符，CloudFront 将根据 RFC 3986 的第 1.2 部分对字符进行百分比编码。

- CloudFront-Viewer-Address – 包含查看器的 IP 地址和请求的源端口。例如，标头值 198.51.100.10:46532 表示查看器的 IP 地址是 198.51.100.10，请求源端口为 46532。
- CloudFront-Viewer-ASN – 包含查看器的自治系统号 (ASN)。

> **Note**
>
> CloudFront-Viewer-Address 和 CloudFront-Viewer-ASN 可以添加到源请求策略中，但不能添加到缓存策略中。

- CloudFront-Viewer-Country – 包含查看器所在国家/地区的双字母国家/地区代码。有关国家/地区代码的列表，请参阅 ISO 3166-1 alpha-2。
- CloudFront-Viewer-City – 包含查看器所在城市的名称。

当您添加以下标头时，CloudFront 会将其应用于除 那些源自 AWS 网络的请求之外的所有请求：

- CloudFront-Viewer-Country-Name – 包含查看器所在国家/地区的名称。
- CloudFront-Viewer-Country-Region – 包含代表查看器所在区域的代码（最多三个字符）。区域是 ISO 3166-2 代码的第一级细分（最广泛或最不具体）。
- CloudFront-Viewer-Country-Region-Name – 包含查看器所在区域的名称。区域是 ISO 3166-2 代码的第一级细分（最广泛或最不具体）。
- CloudFront-Viewer-Latitude – 包含查看器的近似纬度。
- CloudFront-Viewer-Longitude – 包含查看器的近似经度。
- CloudFront-Viewer-Metro-Code – 包含查看器的都市代码。仅当查看器所在地区是美国时提供此项。
- CloudFront-Viewer-Postal-Code – 包含查看器的邮政编码。
- CloudFront-Viewer-Time-Zone 包含查看器的时区，采用 IANA 时区数据库格式（例如，America/Los_Angeles）。

用于确定查看器的标头结构的标头

您可以添加以下标头来帮助根据查看器发送的标头来识别查看器。例如，不同的浏览器可能会按特定顺序发送 HTTP 标头。如果 User-Agent 标头中指定的浏览器与该浏览器的预期标头顺序不匹配，您可以拒绝请求。此外，如果 CloudFront-Viewer-Header-Count 值与 CloudFront-Viewer-Header-Order 中的标头数量不匹配，您可以拒绝请求。

- `CloudFront-Viewer-Header-Order` – 按请求的顺序包含查看器的标头名称（用冒号分隔）。例如：`CloudFront-Viewer-Header-Order: Host:User-Agent:Accept:Accept-Encoding`。超过 7680 字符限制的标头将被截断。
- `CloudFront-Viewer-Header-Count` – 包含查看器标头的总数。

其他 CloudFront 标头

您可以添加以下标头来确定查看器的协议、版本、JA3 指纹和 TLS 连接详细信息：

- `CloudFront-Forwarded-Proto` – 包含查看器请求的协议（HTTP 或 HTTPS）。
- `CloudFront-Viewer-Http-Version` – 包含查看器请求的 HTTP 版本。
- `CloudFront-Viewer-JA3-Fingerprint` – 包含查看器的 JA3 指纹。JA3 指纹可帮助您确定请求是来自已知客户端、恶意软件或恶意机器人还是预期的（允许列出的）应用程序。此标头依赖于查看器的 SSL/TLS `Client Hello` 数据包，仅用于 HTTPS 请求。

  > **Note**
  >
  > 您可以在源请求策略中添加 `CloudFront-Viewer-JA3-Fingerprint`，但不能在缓存策略中添加。

- `CloudFront-Viewer-TLS` – 包含 SSL/TLS 版本、密码以及与查看器和 CloudFront 之间的连接使用的 SSL/TLS 握手相关的信息。标头值采用以下格式：

  ```
  SSL/TLS_version:cipher:handshake_information
  ```

  对于 *`handshake_information`*，标头可包含以下值：

  - `fullHandshake` – 对 SSL/TLS 会话执行了完全握手。
  - `sessionResumed` – 恢复了先前的 SSL/TLS 会话。
  - `connectionReused` – 重复使用了先前的 SSL/TLS 连接。

  下面是此标头的一些示例值：

  ```
  TLSv1.3:TLS_AES_128_GCM_SHA256:sessionResumed
  ```

  ```
  TLSv1.2:ECDHE-ECDSA-AES128-GCM-SHA256:connectionReused
  ```

```
TLSv1.1:ECDHE-RSA-AES128-SHA256:fullHandshake
```

```
TLSv1:ECDHE-RSA-AES256-SHA:fullHandshake
```

有关可能包含在此标头值中的 SSL/TLS 版本和密码的完整列表，请参阅the section called “查看器和 CloudFront 之间支持的协议和密码”。

> **Note**
>
> 您可以在源请求策略中添加 `CloudFront-Viewer-TLS`，但不能在缓存策略中添加。

了解源请求策略和缓存策略如何协同工作

您可以使用 CloudFront 源请求策略来控制 CloudFront 向源发送的请求，这些请求称为源请求。要使用源请求策略，必须将缓存策略附加到相同的缓存行为。如果没有缓存策略，则无法在缓存行为中使用源请求策略。有关更多信息，请参阅 使用策略来控制源请求。

源请求策略和缓存策略协同工作以确定 CloudFront 在源请求中包含的值。您在缓存键中指定的所有 URL 查询字符串、HTTP 标头和 Cookie（使用缓存策略）都将自动包含在源请求中。您在源请求策略中指定的任何其他查询字符串、标头和 Cookie 也都将包含在源请求中（但不会包含在缓存键中）。

源请求策略和缓存策略的设置似乎相互冲突。例如，一个策略可能允许某些值，而另一个策略可能阻止这些值。下表说明了当您同时使用源请求策略和缓存策略的设置时，CloudFront 在源请求中包含哪些值。这些设置通常适用于所有类型的值（查询字符串、标头和 Cookie），唯一的不同是您无法在缓存策略中指定所有标头或使用标头阻止列表。

	源请求策略			
	无	全部	允许列表	阻止列表
缓存策略				
无	除了每个源请求中包含的默认值外，源请求中不包含来自查看器	查看器请求中的所有值都将包含在源请求中。	只有源请求策略中指定的值才包含在源请求中。	除了源请求策略中指定的值，来自查看器请求的

	源请求策略			
	无	全部	允许列表	阻止列表
	请求的任何值。有关更多信息，请参阅 使用策略来控制源请求。			所有值都包含在源请求中。
全部 注意：您不能在缓存策略中指定所有标头。	来自查看器请求的所有查询字符串和 Cookie 都包含在源请求中。	查看器请求中的所有值都包含在源请求中。	来自查看器请求的所有查询字符串和 Cookie 以及源请求策略中指定的任何标头都包含在源请求中。	来自查看器请求的所有查询字符串和 Cookie 都包含在源请求中，即使是源请求策略阻止列表中指定的查询字符串和 Cookie 也是如此。缓存策略设置会覆盖源请求策略阻止列表。
允许列表	只有查看器请求中指定的值会包含在源请求中。	查看器请求中的所有值都包含在源请求中。	在缓存策略或源请求请求策略中指定的所有值都包含在源请求中。	缓存策略中指定的值包含在源请求中，即使在源请求策略阻止列表中指定了相同的值也是如此。缓存策略允许列表将覆盖源请求策略阻止列表。

	源请求策略			
	无	全部	允许列表	阻止列表
阻止列表 注意：您无法在缓存策略阻止列表中指定标头。	除了指定的查询字符串和 Cookie ，来自查看器请求的所有查询字符串和 Cookie 都包含在源请求中。	查看器请求中的所有值都包含在源请求中。	源请求策略中指定的值包含在源请求中，即使在缓存策略阻止列表中指定了相同的值也是如此。源请求策略允许列表将覆盖缓存策略阻止列表。	除了缓存策略或源请求策略中指定的值，来自查看器请求的所有值都包含在源请求中。

使用策略在 CloudFront 响应中添加或删除 HTTP 标头

您可以配置 CloudFront 以修改它发送给查看器（Web 浏览器和其他客户端）的响应中的 HTTP 标头。在将响应发送给查看器之前，CloudFront 可以删除从源接收到的标头，或者在响应中添加标头。进行这些更改不需要编写代码或更改源。

例如，您可以删除 `X-Powered-By` 和 `Vary` 等标头，以便 CloudFront 在发送给查看器的响应中不包含这些标头。或者，您可以添加 HTTP 标头，例如：

- 用于控制浏览器缓存的 `Cache-Control` 标头。
- 用于启用跨源资源共享 (CORS) 的 `Access-Control-Allow-Origin` 标头。您还可以添加其他 CORS 标头。
- 一组常见的安全标头，例如 `Strict-Transport-Security`、`Content-Security-Policy` 和 `X-Frame-Options`。
- `Server-Timing` 标头，可通过 CloudFront 查看与请求和响应的性能和路由相关的信息。

要指定 CloudFront 在 HTTP 响应中添加或删除的标头，您可以使用响应标头策略。您可以将响应标头策略附加到另一个缓存行为，CloudFront 会修改它发送给匹配缓存行为的请求的响应中的标头。CloudFront 修改它从缓存中提供的响应中的标头和它从源转发的标头。如果源响应中包含响应标头策略中添加的一个或多个标头，则此策略可以指定 CloudFront 是使用从源接收的标头，还是用响应标头策略中的标头覆盖该标头。

CloudFront 为常见的使用案例提供了预定义的响应标头策略（称为托管式策略）。您可以使用这些托管式策略，也可以创建您自己的策略。您可以将单个响应标头策略附加到您的 AWS 账户 中多个分配的多个缓存行为。

有关更多信息，请参阅以下主题。

主题

- 了解响应标头策略
- 创建响应标头策略
- 使用托管式响应标头策略

了解响应标头策略

利用响应标头策略，您可以指定 Amazon CloudFront 在它发送给查看器的响应中删除或添加的 HTTP 标头。有关响应标头策略以及使用它们的原因的更多信息，请参阅使用策略添加或删除响应标头。

以下主题解释了响应标头策略中的设置。这些设置分为下面的主题所代表的类别。

主题

- 策略详细信息（元数据）
- CORS 标头
- 安全标头
- 自定义标头
- 删除标头
- Server-Timing 标头

策略详细信息（元数据）

策略详细信息设置包含有关响应标头策略的元数据。

- 名称 – 用于标识响应标头策略的名称。在控制台中，南滨可以使用名称将策略附加到某个缓存行为。
- 描述（可选）– 用于描述响应标头策略的注释。虽然此选项是可选的，但它可以帮助您确定策略的用途。

CORS 标头

跨源资源共享（CORS）设置允许您在响应标头策略中添加和配置 CORS 标头。

此列表关注的是如何在响应标头策略中指定设置和有效值。有关各个标头以及如何在真实的 CORS 请求和响应中使用它们的更多信息，请参阅 MDN Web Docs 中的跨源资源共享和 CORS 协议规范。

Access-Control-Allow-Credentials

这是一个布尔值设置（`true` 或 `false`），它决定了 CloudFront 是否在对 CORS 请求的响应中添加 `Access-Control-Allow-Credentials` 标头。如果此设置设为 `true`，CloudFront 将会在对 CORS 请求的响应中添加 `Access-Control-Allow-Credentials: true` 标头。否则，CloudFront 不会在响应中添加此标头。

Access-Control-Allow-Headers

指定 CloudFront 在 CORS 预检请求的响应中用作 `Access-Control-Allow-Headers` 标头的值的标头名称。此设置的有效值包括 HTTP 标头名称或通配符（`*`）（这表示允许所有标头）。

> **Note**
>
> `Authorization` 标头不能使用通配符，必须明确列出。

通配符的有效使用示例：

示例	将匹配	将不匹配

`x-amz-*`	`x-amz-test`
	
	`x-amz-`
		`x-amz`

`x-*-amz`	`x-test-amz`
	
	`x--amz`	

`*`	除 `Authorization` 外的所有标头	`Authorization`

Access-Control-Allow-Methods

指定 CloudFront 在 CORS 预检请求的响应中用作 `Access-Control-Allow-Methods` 标头的值的 HTTP 方法。有效值为 `GET`、`DELETE`、`HEAD`、`OPTIONS`、`PATCH`、`POST`、`PUT` 或 `ALL`（全部）。`ALL` 是一个包含所有列出的 HTTP 方法的特殊值。

Access-Control-Allow-Origin

指定 CloudFront 可以在 `Access-Control-Allow-Origin` 响应标头中使用的值。此设置的有效值包括特定的源（例如 `http://www.example.com`）或通配符（`*`）（这表示允许所有源）。请参阅下表中的示例：

> **Note**
>
> 允许将通配符（`*`）用作域（`*.example.org`）的最左侧部分。

不 允许在以下位置使用通配符（*）：

- 顶级域（example.*）
- 子域（test.*.example.org）的右侧
- 术语含义（exa*mple.org））

下表显示了通配符的有效使用示例：

示例	将匹配	将不匹配
http://*.example.org	http://www.example.org http://test.example.org http://test.example.org:123	**https**://test.example.org **https**://test.example.org:123
*.example.org	test.example.org test.test.example.org .example.org http://test.example.org https://test.example.org http://test.example.org:123 https://test.example.org:123	
example.org	http://example.org	

示例	将匹配	将不匹配
	`https://example.org`	
`http://example.org`		**`https`**`://example.org` `http://example.org:123`

`http://example.org:*`	`http://example.org:123`
	
	`http://example.org`	

`http://example.org:1*3`	`http://example.org:123`
	
	`http://example.org:1893`
	
	`http://example.org:13`	

`*.example.org:1*`	`test.example.org:123`	

Access-Control-Expose-Headers

指定 CloudFront 在 CORS 请求的响应中用作 `Access-Control-Expose-Headers` 标头的值的标头名称。此设置的有效值包括 HTTP 标头名称或通配符（`*`）。

Access-Control-Max-Age

CloudFront 在 CORS 预检请求的响应中用作 `Access-Control-Max-Age` 标头的值的秒数。

源覆盖

一个布尔值设置，它决定了当来自源的响应包含策略中也包含的某个 CORS 标头时 CloudFront 的行为方式。

- 如果设置为 `true` 并且源响应包含策略中也包含的 CORS 标头，CloudFront 会将策略中的 CORS 标头添加到该响应中。然后，CloudFront 会将该响应发送给查看器。CloudFront 会忽略从源收到的标头。

- 如果设置为 `false` 并且源响应包含 CORS 标头（无论策略中是否包含该 CORS 标头），CloudFront 都会将它从源收到的 CORS 标头添加到响应中。CloudFront 不会将策略中的任何 CORS 标头添加到发送给查看器的响应中。

安全标头

可以使用安全标头设置在响应标头策略中添加和配置多个与安全相关的 HTTP 响应标头。

此列表描述如何在响应标头策略中指定设置和有效值。有关各个标头以及如何在真实的 HTTP 响应中使用它们的更多信息，请参阅 MDN Web Docs 链接。

Content-Security-Policy

指定 CloudFront 用作 `Content-Security-Policy` 响应标头的值的内容安全策略指令。

有关此标头和有效策略指引的更多信息，请参阅 MDN Web Docs 中的 Content-Security-Policy。

> **Note**
>
> `Content-Security-Policy` 标头值的长度限制为 1783 个字符。

Referrer-Policy

指定 CloudFront 用作 `Referrer-Policy` 响应标头的值的引用站点策略指令。此设置的有效值为 `no-referrer`、`no-referrer-when-downgrade`、`origin`、`origin-when-cross-origin`、`same-origin`、`strict-origin`、`strict-origin-when-cross-origin` 和 `unsafe-url`。

有关此标头和这些指引的更多信息，请参阅 MDN Web Docs 中的 Referrer-Policy。

Strict-Transport-Security

指定 CloudFront 用作 `Strict-Transport-Security` 响应标头的值的指令和设置。对于此设置，您可以单独指定：

- CloudFront 用作此标头的 `max-age` 指令的值的秒数
- `preload` 的布尔值设置（`true` 或 `false`），它决定了 CloudFront 是否在此标头的值中包括 `preload` 指令
- `includeSubDomains` 的布尔值设置（`true` 或 `false`），它决定了 CloudFront 是否在此标头的值中包括 `includeSubDomains` 指令

有关此标头和这些指引的更多信息，请参阅 MDN Web Docs 中的 Strict-Transport-Security。

X-Content-Type-Options

这是一个布尔值设置（`true` 或 `false`），它决定了 CloudFront 是否在响应中添加 `X-Content-Type-Options` 标头。如果此设置为 `true`，CloudFront 会在响应中添加 `X-Content-Type-Options: nosniff` 标头。否则，CloudFront 不会添加此标头。

有关此标头的更多信息，请参阅 MDN Web Docs 中的 X-Content-Type-Options。

X-Frame-Options

指定 CloudFront 用作 `X-Frame-Options` 响应标头的值的指令。此设置的有效值为 `DENY` 或 `SAMEORIGIN`。

有关此标头和这些指引的更多信息，请参阅 MDN Web Docs 中的 X-Frame-Options。

X-XSS-Protection

指定 CloudFront 用作 `X-XSS-Protection` 响应标头的值的指令和设置。对于此设置，您可以单独指定：

- `X-XSS-Protection` 设置为 `0`（禁用 XSS 筛选）或 `1`（启用 XSS 筛选）
- `block` 的布尔值设置（`true` 或 `false`），它决定了 CloudFront 是否在此标头的值中包括 `mode=block` 指令
- 报告 URI，它决定了 CloudFront 是否在此标头的值中包括 `report=`*`reporting URI`* 指令

您可以为 `block` 指定 `true`，也可以指定一个报告 URI，但是不能同时指定这两者。有关此标头和这些指引的更多信息，请参阅 MDN Web Docs 中的 X-XSS-Protection。

源覆盖

这些安全标头设置均包含一个布尔值设置（`true` 或 `false`），它决定了当来自源的响应中包含该标头时 CloudFront 的行为方式。

如果此设置设为 `true` 并且源响应中包含此标头，CloudFront 会将策略中的标头添加到它发送给查看器的响应中。它忽略从源收到的标头。

如果此设置设为 `false` 并且源响应中包含此标头，CloudFront 会将它从源收到的标头包含在它发送给查看器的响应中。

当源响应不包含此标头时，CloudFront 会将策略中的标头添加到它发送给查看器的响应中。当此设置设为 `true` 或 `false` 时，CloudFront 都会执行此操作。

自定义标头

您可以使用自定义标头设置在响应标头策略中添加和配置自定义的 HTTP 标头。CloudFront 会将这些标头添加到它返回给查看器的每个响应中。对于每个自定义标头，您还可以指定标头的值，但指定值的步骤是可选的。这是因为 CloudFront 可以添加一个没有值的响应标头。

每个自定义标头也有自身的源覆盖设置：

- 如果此设置设为 `true` 并且源响应中包含策略中也包含的自定义标头，CloudFront 会将策略中的自定义标头添加到它发送给查看器的响应中。它忽略从源收到的标头。
- 如果此设置为 `false` 并且源响应中包含策略中也包含的自定义标头，CloudFront 会将它从源收到的自定义标头包含在它发送给查看器的响应中。
- 如果源响应中不包含策略中所包含的自定义标头，CloudFront 会将策略中的自定义标头添加到它发送给查看器的响应中。当此设置设为 `true` 或 `false` 时，CloudFront 都会执行此操作。

删除标头

您可以指定您希望 CloudFront 在它从源收到的响应中删除的标头，以使这些标头不包含在 CloudFront 发送给查看器的响应中。无论对象是从 CloudFront 的缓存还是源提供的，CloudFront 都会从发送给查看器的每个响应中删除标头。例如，您可以删除对浏览器无用的标头（例如 `X-Powered-By` 或 `Vary`），以便 CloudFront 从它发送给查看器的响应中删除这些标头。

当您使用响应标头策略指定要删除的标头时，CloudFront 会先删除标头，然后添加响应标头策略的其他部分中指定的任何标头（CORS 标头、安全标头、自定义标头等）。如果您指定要删除的标头，并在策略的另一个部分中添加相同的标头，则 CloudFront 会在发送给查看器的响应中包含该标头。

> **Note**
>
> 您可以使用响应标头策略删除 CloudFront 从源接收的 `Server` 和 `Date` 标头，以使 CloudFront 发送给查看器的响应中不包含这些标头（从源接收的）。但如果您这样做，CloudFront 会将其拥有的这些标头的版本添加到发送给查看器的响应中。对于 CloudFront 添加的 `Server` 标头，标头的值为 `CloudFront`。

您无法删除的标头

您无法使用响应标头策略删除以下标头。如果您在响应标头策略（API 中的 ResponseHeadersPolicyRemoveHeadersConfig）的删除标头部分指定这些标头，则会收到错误。

- Connection
- Content-Encoding
- Content-Length
- Expect
- Host
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- Proxy-Connection
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Warning
- X-Accel-Buffering
- X-Accel-Charset
- X-Accel-Limit-Rate
- X-Accel-Redirect
- X-Amz-Cf-.*
- X-Amzn-Auth
- X-Amzn-Cf-Billing
- X-Amzn-Cf-Id
- X-Amzn-Cf-Xff
- X-Amzn-ErrorType
- X-Amzn-Fle-Profile

- X-Amzn-Header-Count
- X-Amzn-Header-Order
- X-Amzn-Lambda-Integration-Tag
- X-Amzn-RequestId
- X-Cache
- X-Edge-.*
- X-Forwarded-Proto
- X-Real-Ip

Server-Timing 标头

使用 Server-Timing 标头设置在从 CloudFront 发送的 HTTP 响应中启用 Server-Timing 标头。您可以使用此标头查看指标，这些指标可以帮助您深入了解 CloudFront 和源的行为和性能。例如，您可以看到哪个缓存层提供了缓存命中。或者，如果存在缓存未命中，您可以看到源中的第一个字节延迟。Server-Timing 标头中的指标可以帮助您排查问题或测试 CloudFront 或源配置的效率。

有关结合使用 Server-Timing 标头与 CloudFront 的更多信息，请参阅以下主题。

要启用 Server-Timing 标头，请创建（或编辑）响应标头策略。

主题

- 采样率和 Pragma 请求标头
- 源中的 Server-Timing 标头
- Server-Timing 标头指标
- Server-Timing 标头示例

采样率和 Pragma 请求标头

当您在响应标头策略中启用 Server-Timing 时，还可以指定采样率。采样率是 0–100（含）之间的一个数字，它指定您希望 CloudFront 将 Server-Timing 标头添加到其中的响应的百分比。当您将采样率设置为 100 时，CloudFront 会针对每个与响应标头策略附加到的缓存行为相匹配的请求，将 Server-Timing 标头添加到对应的 HTTP 响应中。当您将其设置为 50 时，CloudFront 会将此标头添加到与缓存行为匹配的请求的 50% 的响应中。您可以将采样率设置为 0–100 之间的任意数字，最多保留四位小数。

当采样率设置为低于 100 的数字时，您无法控制 CloudFront 将 Server-Timing 标头添加到哪些响应中，只能控制百分比。但是，您可以在 HTTP 请求中添加值设置为 server-timing 的 Pragma 标头，以便在对该请求的响应中接收 Server-Timing 标头。无论采样率设置为多少，这一点都适用。即使采样率设为零（0），如果请求包含 Pragma: server-timing 标头，CloudFront 也会将 Server-Timing 标头添加到响应中。

源中的 Server-Timing 标头

当缓存未命中且 CloudFront 将请求转发到源时，源可能会在至 CloudFront 的响应中包含 Server-Timing 标头。在这种情况下，CloudFront 将其指标添加到它从源接收到的 Server-Timing 标头中。CloudFront 发送给查看器的响应包含一个 Server-Timing 标头，其中包含来自源的值和 CloudFront 添加的指标。源中的标头值可能位于末尾，或者介于 CloudFront 添加到标头的两组指标之间。

当出现缓存命中时，CloudFront 发送给查看器的响应将包含一个 Server-Timing 标头，其中的标头值中仅包含 CloudFront 指标（不包括源中的值）。

Server-Timing 标头指标

当 CloudFront 将 Server-Timing 标头添加到 HTTP 响应时，标头的值包含一个或多个指标，它们可以帮助您深入了解 CloudFront 的行为和性能。以下列表包含所有指标及其可能的值。Server-Timing 标头仅包含其中一些指标，具体取决于通过 CloudFront 的请求和响应的性质。

其中一些指标包含在仅具有名称（而没有值）的 Server-Timing 标头中。其他指标则为名称和值。当指标有值时，名称和值由分号（;）分隔。当标头包含多个指标时，指标用逗号（,）分隔。

cdn-cache-hit

CloudFront 在未向源发出请求的情况下提供了来自缓存的响应。

cdn-cache-refresh

CloudFront 在向源发送请求以验证缓存对象仍然有效后提供了来自缓存的响应。在这种情况下，CloudFront 不会从源中检索完整对象。

cdn-cache-miss

CloudFront 未提供来自缓存的响应。在这种情况下，CloudFront 在返回响应之前从源中请求了完整对象。

cdn-pop

包含一个值，此值描述了是哪个 CloudFront 入网点（POP）处理了请求。

cdn-rid

包含一个值，表示该请求的 CloudFront 唯一标识符。当通过 AWS Support 排查问题时，可以使用此请求标识符（RID）

cdn-hit-layer

当 CloudFront 从缓存中提供响应而不向源发出请求时，此指标存在。它包含下列值之一：

- EDGE – CloudFront 提供了来自 POP 位置的缓存响应。
- REC – CloudFront 提供了来自区域边缘缓存（REC）位置的缓存响应。
- Origin Shield – CloudFront 提供了来自充当 Origin Shield 的 REC 的缓存响应。

cdn-upstream-layer

当 CloudFront 从源请求完整对象时，此指标存在并包含下列值之一：

- EDGE – POP 位置已将请求直接发送到源。
- REC – REC 位置已将请求直接发送到源。
- Origin Shield – 充当 Origin Shield 的 REC 已将请求直接发送到源。

cdn-upstream-dns

包含一个值，表示检索源的 DNS 记录所花费的毫秒数。值为零（0）表示 CloudFront 使用了缓存的 DNS 结果或重用了现有连接。

cdn-upstream-connect

包含一个值，表示源 DNS 请求完成与到源的 TCP（和 TLS，如果适用）连接完成之间的毫秒数。值为零（0）表示 CloudFront 重用了现有连接。

cdn-upstream-fbl

包含一个值，表示从源 HTTP 请求完成到从来自源的响应中收到第一个字节之间的毫秒数（第一个字节延迟）。

cdn-downstream-fbl

包含一个值，表示边缘站点完成接收请求的时间与它将响应的首个字节发送至查看器的时间之间的毫秒数。

Server-Timing 标头示例

以下是启用 `Server-Timing` 标头设置时，查看器可能从 CloudFront 接收的 `Server-Timing` 标头的示例。

Example – 缓存未命中

以下示例显示当请求的对象不在 CloudFront 缓存中时，查看器可能会收到的 Server-Timing 标头。

```
Server-Timing: cdn-upstream-layer;desc="EDGE",cdn-upstream-dns;dur=0,cdn-upstream-
connect;dur=114,cdn-upstream-fbl;dur=177,cdn-cache-miss,cdn-pop;desc="PHX50-C2",cdn-
rid;desc="yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg==",cdn-downstream-
fbl;dur=436
```

此 Server-Timing 标头表示以下内容：

- 源请求是从 CloudFront 入网点（POP）位置（cdn-upstream-layer;desc="EDGE"）发送的。
- CloudFront 为源使用了缓存的 DNS 结果（cdn-upstream-dns;dur=0）。
- CloudFront 花了 114 毫秒完成了与源的 TCP（和 TLS，如果适用）连接（cdn-upstream-connect;dur=114）。
- 在完成请求之后，CloudFront 花了 177 毫秒收到来自源的响应的第一个字节（cdn-upstream-fbl;dur=177）。
- 请求的对象不在 CloudFront 的缓存中 (cdn-cache-miss)。
- 请求是在由代码标识的边缘站点收到的 PHX50-C2 (cdn-pop;desc="PHX50-C2")。
- 该请求的 CloudFront 唯一 ID 是 yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg== (cdn-rid;desc="yNPsyYn7skvTzwWkq3Wcc8Nj_foxUjQUe9H1ifslzWhb0w7aLbFvGg==")。
- 在收到查看器请求后，CloudFront 花了 436 毫秒才将响应的第一个字节发送到查看器（cdn-downstream-fbl;dur=436）。

Example – 缓存命中

以下示例显示当请求的对象在 CloudFront 的缓存中时，查看器可能会收到的 Server-Timing 标头。

```
Server-Timing: cdn-cache-hit,cdn-pop;desc="SEA19-C1",cdn-
rid;desc="nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g==",cdn-hit-
layer;desc="REC",cdn-downstream-fbl;dur=137
```

此 Server-Timing 标头表示以下内容：

- 请求的对象在缓存中 (cdn-cache-hit)。
- 请求是在由代码标识的边缘站点收到的 SEA19-C1 (cdn-pop;desc="SEA19-C1")。

- 该请求的 CloudFront 唯一 ID 是 nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g== (cdn-rid;desc="nQBz4aJU2kP9iC3KHEq7vFxfMozu-VYBwGzkW9diOpeVc7xsrLKj-g==")。
- 请求的对象已缓存在区域边缘缓存 (REC) 位置 (cdn-hit-layer;desc="REC")。
- 在收到查看器请求后，CloudFront 花了 137 毫秒才将响应的第一个字节发送到查看器（cdn-downstream-fbl;dur=137）。

创建响应标头策略

可以使用响应标头策略指定 Amazon CloudFront 在 HTTP 响应中添加或删除的 HTTP 标头。有关响应标头策略以及使用它们的原因的更多信息，请参阅使用策略添加或删除响应标头。

您可以在 CloudFront 控制台中创建响应标头策略。或者，您可以使用 AWS CloudFormation、AWS Command Line Interface（AWS CLI）或 CloudFront API 创建一个标头。创建响应标头策略后，将它附加到 CloudFront 分配中的一个或多个缓存行为。

在创建自定义响应标头策略之前，检查是否有某个托管式响应标头策略适合您的使用案例。如果有，则可以将它附加到缓存行为。这样，您就不需要创建或管理自己的响应标头策略。

Console

创建响应标头策略（控制台）

1. 登录到AWS Management Console，打开 CloudFront 控制台（地址为 https://console.aws.amazon.com/cloudfront/v4/home#/policies/responseHeaders）的 Policies（策略）页面，转到 Response headers（响应标头）选项卡。
2. 选择 Create response headers policy（创建响应标头策略）。
3. 在 Create response headers policy（创建响应标头策略）表单中，执行以下操作：

   a. 在 Details（详细信息）面板中，输入响应标头策略的 Name（名称）和（可选）解释策略用途的 Description（描述）。

   b. 在 Cross-origin resource sharing (CORS) [跨源资源共享（CORS）]面板中，选择 Configure CORS（配置 CORS）切换按钮并配置要添加到策略的任何 CORS 标头。如果您希望配置的标头覆盖 CloudFront 从源接收的标头，请选中 Origin override（源覆盖）复选框。

      有关 CORS 标头设置的更多信息，请参阅the section called “CORS 标头”。

c. 在 Security headers（安全标头）面板中，选择切换按钮并配置要添加到策略的每个安全标头。

有关安全标头设置的更多信息，请参阅the section called “安全标头”。

d. 在 Custom headers（自定义标头）面板中，添加要包含在策略中的任何自定义标头。

有关自定义标头设置的更多信息，请参阅the section called “自定义标头”。

e. 在 Remove headers（删除标头）面板中，添加您希望 CloudFront 从源响应中删除的且不包含在 CloudFront 发送给查看器的响应中的所有标头名称。

有关删除标头设置的更多信息，请参阅the section called “删除标头”。

f. 在 Server-Timing header（Server-Timing 标头）面板中，选择 Enable（启用）开关并输入采样率（0 到 100（含）之间的数字）。

有关 `Server-Timing` 标头的更多信息，请参阅the section called “Server-Timing 标头”。

4. 选择 Create（创建）以创建策略。

创建响应标头策略后，您可以将它附加到 CloudFront 分配中的某个缓存行为。

将响应标头策略附加到现有分配（控制台）

1. 在 CloudFront 控制台中打开 Distributions (分配) 页面，网址为 https://console.aws.amazon.com/cloudfront/v4/home#/distributions。
2. 选择要更新的分配，然后选择行为选项卡。
3. 选择要更新的缓存行为，然后选择 Edit（编辑）。

或者，要创建新的缓存行为，请选择 Create behavior(创建行为)。

4. 对于 Response headers policy（响应标头策略），选择要添加到缓存行为的策略。
5. 选择 Save changes（保存更改）以更新缓存行为。如果要创建新的缓存行为，请选择 Create behavior（创建行为）。

将响应标头策略附加到新的分配（控制台）

1. 通过 打开 CloudFront 控制台https://console.aws.amazon.com/cloudfront/v4/home
2. 选择 Create distribution（创建分配）。

3. 对于 Response headers policy（响应标头策略），选择要添加到缓存行为的策略。
4. 选择分配的其他设置。有关更多信息，请参阅 the section called “分配设置”。
5. 选择 Create distribution（创建分配）以创建分配。

AWS CloudFormation

要使用 AWS CloudFormation 创建响应标头策略，请使用 `AWS::CloudFront::ResponseHeadersPolicy` 资源类型。以下示例显示了 YAML 格式的 AWS CloudFormation 模板语法，用于创建响应标头策略。

```
Type: AWS::CloudFront::ResponseHeadersPolicy
Properties:
  ResponseHeadersPolicyConfig:
    Name: EXAMPLE-Response-Headers-Policy
    Comment: Example response headers policy for the documentation
    CorsConfig:
      AccessControlAllowCredentials: false
      AccessControlAllowHeaders:
        Items:
          - '*'
      AccessControlAllowMethods:
        Items:
          - GET
          - OPTIONS
      AccessControlAllowOrigins:
        Items:
          - https://example.com
          - https://docs.example.com
      AccessControlExposeHeaders:
        Items:
          - '*'
      AccessControlMaxAgeSec: 600
      OriginOverride: false
    CustomHeadersConfig:
      Items:
        - Header: Example-Custom-Header-1
          Value: value-1
          Override: true
        - Header: Example-Custom-Header-2
          Value: value-2
          Override: true
    SecurityHeadersConfig:
```

```
      ContentSecurityPolicy:
        ContentSecurityPolicy: default-src 'none'; img-src 'self'; script-src
 'self'; style-src 'self'; object-src 'none'; frame-ancestors 'none'
        Override: false
      ContentTypeOptions: # You don't need to specify a value for 'X-Content-Type-
Options'.
                          # Simply including it in the template sets its value to
 'nosniff'.
        Override: false
      FrameOptions:
        FrameOption: DENY
        Override: false
      ReferrerPolicy:
        ReferrerPolicy: same-origin
        Override: false
      StrictTransportSecurity:
        AccessControlMaxAgeSec: 63072000
        IncludeSubdomains: true
        Preload: true
        Override: false
      XSSProtection:
        ModeBlock: true # You can set ModeBlock to 'true' OR set a value for
 ReportUri, but not both
        Protection: true
        Override: false
    ServerTimingHeadersConfig:
      Enabled: true
      SamplingRate: 50
    RemoveHeadersConfig:
      Items:
        - Header: Vary
        - Header: X-Powered-By
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 AWS::CloudFront::ResponseHeadersPolicy。

CLI

要使用 AWS Command Line Interface（AWS CLI）创建响应标头策略，请使用 aws cloudfront create-response-headers-policy 命令。您可以使用输入文件来提供命令的输入参数，而不是将每个单独的参数指定为命令行输入。

创建响应标头策略（通过 CLI 并使用输入文件）

1. 使用以下命令创建名为 `response-headers-policy.yaml` 的文件。此文件包含 create-response-headers-policy 命令的所有输入参数。

```
aws cloudfront create-response-headers-policy --generate-cli-skeleton yaml-input
 > response-headers-policy.yaml
```

2. 打开刚创建的 `response-headers-policy.yaml` 文件。编辑文件以指定策略名称和所需的响应标头策略配置，然后保存文件。

   有关响应标头策略设置的更多信息，请参阅the section called “了解响应标头策略”。

3. 使用以下命令创建响应标头策略。您创建的策略使用来自 `response-headers-policy.yaml` 文件的输出参数。

```
aws cloudfront create-response-headers-policy --cli-input-yaml file://response-
headers-policy.yaml
```

   记下命令输出中的 `Id` 值。这是响应标头策略 ID。您需要它才能将策略附加到 CloudFront 分配的缓存行为。

将响应标头策略附加到现有分配（通过 CLI 并使用输入文件）

1. 使用以下命令保存要更新的 CloudFront 分配的分配配置。将 *`distribution_ID`* 替换为分配 ID。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml >
 dist-config.yaml
```

2. 打开刚创建的名为 `dist-config.yaml` 的文件。编辑此文件，对缓存行为进行以下更改以使它使用响应标头策略。

   - 在缓存行为中，添加名为 `ResponseHeadersPolicyId` 的字段。对于字段的值，请使用创建策略后记下的响应标头策略 ID。
   - 将 `ETag` 字段重命名为 `IfMatch`，但不更改字段的值。

完成后保存该文件。

3. 使用以下命令将分配更新为使用响应标头策略。将 *distribution_ID* 替换为分配 ID。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

将响应标头策略附加到新的分配（通过 CLI 并使用输入文件）

1. 使用以下命令创建名为 `distribution.yaml` 的文件。此文件包含 create-distribution 命令的所有输入参数。

```
aws cloudfront create-distribution --generate-cli-skeleton yaml-input >
 distribution.yaml
```

2. 打开刚创建的 `distribution.yaml` 文件。在默认缓存行为中，在 `ResponseHeadersPolicyId` 字段中输入创建策略后记下的响应标头策略 ID。继续编辑该文件以指定所需的分配设置，然后在完成后保存该文件。

   有关分配设置的更多信息，请参阅分配设置参考。

3. 使用以下命令通过 `distribution.yaml` 文件中的输入参数创建分配。

```
aws cloudfront create-distribution --cli-input-yaml file://distribution.yaml
```

API

要使用 CloudFront API 创建响应标头策略，请使用 CreateResponseHeadersPolicy。有关您在此 API 调用中指定的字段的更多信息，请参阅the section called “了解响应标头策略”以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

创建响应标头策略后，可以使用下面的任何一个 API 调用将其附加到缓存行为：

- 要将该配置附加到现有分配中的缓存行为，请使用 UpdateDistribution。
- 要将该配置附加到新分配中的缓存行为，请使用 CreateDistribution。

对于这两个 API 调用，请在缓存行为内的 `ResponseHeadersPolicyId` 字段中提供响应标头策略 ID。有关您在这些 API 调用中指定的其他字段的更多信息，请参阅 分配设置参考 以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

使用托管式响应标头策略

借助 CloudFront 响应标头策略，您可以指定 Amazon CloudFront 将在它发送给查看器的响应中删除或添加的 HTTP 标头。有关响应标头策略以及使用它们的原因的更多信息，请参阅使用策略添加或删除响应标头。

CloudFront 提供的托管式响应标头策略可以附加到 CloudFront 分配中的缓存行为。借助托管式响应标头策略，您无需编写或维护自己的策略。托管式策略包含针对常见使用案例的一组 HTTP 响应标头。

要使用托管式响应标头策略，您需要将它附加到分配中的缓存行为。此过程与创建自定义响应标头策略的过程相同。但是，您不需要创建新策略，只需附加其中一个托管式策略即可。您可以按名称（使用控制台）或 ID（使用 AWS CloudFormation、AWS CLI或AWS开发工具包）来附加策略。以下部分列出了名称和 ID。

有关更多信息，请参阅 the section called “创建响应标头策略”。

下面的主题介绍了您可以使用的托管响应标头策略。

主题

- CORS-and-SecurityHeadersPolicy
- CORS-With-Preflight
- CORS-with-preflight-and-SecurityHeadersPolicy
- SecurityHeadersPolicy
- SimpleCORS

CORS-and-SecurityHeadersPolicy

在 CloudFront 控制台中查看此策略

使用此托管式策略以允许来自任何源的简单 CORS 请求。此策略还会在 CloudFront 发送给查看器的所有响应中添加一组安全标头。此策略将 the section called “SimpleCORS” 和 the section called “SecurityHeadersPolicy” 策略组合为一个策略。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`e61eb60c-9c35-4d20-a928-2b84e02af89c`

策略设置

	标头名称	标头值	覆盖源？
CORS 标头：	`Access-Control-Allow-Origin`	`*`	否
安全标头：	`Referrer-Policy`	`strict-origin-when-cross-origin`	否
	`Strict-Transport-Security`	`max-age=31536000`	否
	`X-Content-Type-Options`	`nosniff`	是
	`X-Frame-Options`	`SAMEORIGIN`	否
	`X-XSS-Protection`	`1; mode=block`	否

CORS-With-Preflight

在 CloudFront 控制台中查看此策略

使用此托管式策略以允许来自任何源的 CORS 请求，包括预检请求。对于预检请求（使用 HTTP `OPTIONS` 方法），CloudFront 会将下面的三个标头全部添加到响应中。对于简单 CORS 请求，CloudFront 将只添加 `Access-Control-Allow-Origin` 标头。

如果 CloudFront 从源收到的响应包括其中的任何标头，CloudFront 将在它对查看器的响应中使用收到的标头（及其值）。CloudFront 在此策略中不使用标头。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`5cc3b908-e619-4b99-88e5-2cf7f45965bd`

策略设置

	标头名称	标头值	覆盖源？
CORS 标头：	Access-Control-Allow-Methods	DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT	否
	Access-Control-Allow-Origin	*	
	Access-Control-Expose-Headers	*	

CORS-with-preflight-and-SecurityHeadersPolicy

在 CloudFront 控制台中查看此策略

使用此托管式策略以允许来自任何源的 CORS 请求。这包括预检请求。此策略还会在 CloudFront 发送给查看器的所有响应中添加一组安全标头。此策略将 the section called “CORS-With-Preflight” 和 the section called “SecurityHeadersPolicy” 策略组合为一个策略。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

eaab4381-ed33-4a86-88ca-d9558dc6cd63

策略设置

	标头名称	标头值	覆盖源？
CORS 标头：	Access-Control-Allow-Methods	DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT	否
	Access-Control-Allow-Origin	*	
	Access-Control-Expose-Headers	*	

	标头名称	标头值	覆盖源？
安全标头：	Referrer-Policy	strict-origin-when-cross-origin	否
	Strict-Transport-Security	max-age=31536000	否
	X-Content-Type-Options	nosniff	是
	X-Frame-Options	SAMEORIGIN	否
	X-XSS-Protection	1; mode=block	否

SecurityHeadersPolicy

在 CloudFront 控制台中查看此策略

使用此托管式策略以在 CloudFront 发送给查看器的所有响应中添加一组安全标头。有关这些安全标头的更多信息，请参阅 Mozilla 的 Web 安全指南。

借助此响应标头策略，CloudFront 将 `X-Content-Type-Options: nosniff` 添加到所有响应中。当 CloudFront 从源收到的响应包含或不包含此标头时，都是这种情况。对于此策略中的所有其他标头，如果 CloudFront 从源收到的响应包括该标头，CloudFront 将在它对查看器的响应中使用收到的标头（及其值）。它在此策略中不使用标头。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`67f7725c-6f97-4210-82d7-5512b31e9d03`

策略设置

	标头名称	标头值	覆盖源？
安全标头：	Referrer-Policy	strict-origin-when-cross-origin	否

	标头名称	标头值	覆盖源？
	Strict-Transport-Security	max-age=31536000	否
	X-Content-Type-Options	nosniff	是
	X-Frame-Options	SAMEORIGIN	否
	X-XSS-Protection	1; mode=block	否

SimpleCORS

在 CloudFront 控制台中查看此策略

使用此托管式策略以允许来自任何源的简单 CORS 请求。使用此策略时，CloudFront 会在简单 CORS 请求的所有响应中添加标头 `Access-Control-Allow-Origin: *`。

如果 CloudFront 从源收到的响应包括 `Access-Control-Allow-Origin` 标头，CloudFront 将在它对查看器的响应中使用该标头（及其值）。CloudFront 在此策略中不使用标头。

在使用 AWS CloudFormation、AWS CLI 或 CloudFront API 时，此策略的 ID 为：

`60669652-455b-4ae9-85a4-c4c02393f86c`

策略设置

	标头名称	标头值	覆盖源？
CORS 标头：	Access-Control-Allow-Origin	*	否

请求和响应行为

下面各节介绍了 CloudFront 如何处理查看器请求及如何将请求转发给您的 Amazon S3 或自定义源，以及 CloudFront 如何处理来自源的响应，包括 CloudFront 如何处理与缓存 4xx 和 5xx HTTP 状态代码。

主题

CloudFront 如何处理 HTTP 和 HTTPS 请求

对于 Amazon S3 源，默认情况下，CloudFront 接受 CloudFront 分配中对象 HTTP 和 HTTPS 协议的请求。CloudFront 然后使用与发出请求所用的相同协议将请求转发到 Amazon S3 存储桶。

对于自定义源，当您创建分配时，可指定 CloudFront 如何访问源：仅 HTTP，或者匹配查看器使用的协议。有关 CloudFront 如何处理自定义源的 HTTP 和 HTTPS 请求的更多信息，请参阅协议。

有关如何对您的分配进行限制以使最终用户只能使用 HTTPS 访问对象的信息，请参阅将 HTTPS 与 CloudFront 结合使用。

> **Note**
>
> HTTPS 请求的费用高于 HTTP 请求的费用。有关计费费率的更多信息，请参阅 CloudFront 定价。

Amazon S3 源的请求和响应行为

要了解在使用 Amazon S3 作为源时 CloudFront 如何处理请求和响应，请参阅以下部分：

主题

CloudFront 如何处理请求并将请求转发到您的 Amazon S3 源

了解 CloudFront 如何处理查看器请求以及如何将请求转发到您的 Amazon S3 源。

目录

缓存持续时间和最小 TTL

要控制对象在 CloudFront 缓存中保留多长时间后 CloudFront 便会将另一请求转发到您的源，您可以：

- 配置您的源，以添加 `Cache-Control` 或 `Expires` 标题字段到每个对象。

- 在 CloudFront 缓存行为中指定最短 TTL 的值。
- 使用默认值 24 小时。

有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

客户端 IP 地址

如果查看器将请求发送到 CloudFront 且不包含 X-Forwarded-For 请求标头，则 CloudFront 将从 TCP 连接获取查看器的 IP 地址，添加包含该 IP 地址的 X-Forwarded-For 标头，并将请求转发到源。例如，如果 CloudFront 从 TCP 连接获取的 IP 地址为 192.0.2.2，则它会将以下标头转发到源：

```
X-Forwarded-For: 192.0.2.2
```

如果查看器将请求发送到 CloudFront 且包含 X-Forwarded-For 请求标头，则 CloudFront 将从 TCP 连接获取查看器的 IP 地址，将该 IP 地址附加到 X-Forwarded-For 标头的末尾，并将请求转发到源。例如，如果查看器请求包含 X-Forwarded-For: 192.0.2.4,192.0.2.3，并且 CloudFront 从 TCP 连接获取的 IP 地址为 192.0.2.2，则它会将以下标头转发到源：

```
X-Forwarded-For: 192.0.2.4,192.0.2.3,192.0.2.2
```

> **Note**
>
> X-Forwarded-For 标头包含 IPv4 地址（例如 192.0.2.44）和 IPv6 地址（例如 2001:0db8:85a3::8a2e:0370:7334）。

有条件 GET 请求

当 CloudFront 收到的请求针对的是边缘缓存中已过期的对象时，它将请求转发到 Amazon S3 源，以获取对象的最新版本，或者从 Amazon S3 获取 CloudFront 边缘缓存已有最新版本的确认。当 Amazon S3 最初发送对象到 CloudFront 时，其在响应中包括 ETag 值和 LastModified 值。在 CloudFront 转发到 Amazon S3 的新请求中，CloudFront 添加以下一个或两个标头：

- If-Match 或 If-None-Match 标头，其中包括对象过期版本的 ETag 值。
- If-Modified-Since 标头，其中包含对象过期版本的 LastModified 值。

Amazon S3 使用此信息来确定对象是否已更新，以及是否将整个对象返回到 CloudFront 或只返回 HTTP 304 状态码（未修改）。

Cookie

Amazon S3 不处理 Cookie。如果您配置缓存行为，以将 cookie 转发到 Amazon S3 源，CloudFront 将转发 cookie 而 Amazon S3 则会忽略他们。同一对象的所有将来请求 (无论您是否更改 Cookie) 都将通过缓存中的现有对象提供。

跨源资源共享 (CORS)

如果您希望 CloudFront 尊重 Amazon S3 跨源资源共享设置，请将 CloudFront 配置为将所选标头转发到 Amazon S3。有关更多信息，请参阅 根据请求标头缓存内容。

包含正文的 GET 请求

如果查看器 GET 请求包含正文，则 CloudFront 将 HTTP 状态代码 403（禁止）返回给查看器。

HTTP 方法

如果您将 CloudFront 配置为处理其支持的所有 HTTP 方法，则 CloudFront 会接受来自查看器的以下请求，并将这些请求转发给您的 Amazon S3 源：

- DELETE
- GET
- HEAD
- OPTIONS
- PATCH
- POST
- PUT

CloudFront 始终缓存对 GET 和 HEAD 请求的响应。您也可以将 CloudFront 配置为缓存对 OPTIONS 请求的响应。CloudFront 不缓存对使用其他方法的请求的响应。

如果您想使用多部分上传方式将对象添加到 Amazon S3 存储桶，则必须将 CloudFront 源访问控制（OAC）添加到您的分配中，并向 OAC 授予所需的许可。有关更多信息，请参阅 the section called “限制对 Amazon Simple Storage Service 源的访问”。

> **Important**
>
> 如果您将 CloudFront 配置为接受 CloudFront 支持的所有 HTTP 方法并将这些方法转发到 Amazon S3，则必须创建一个 CloudFront OAC 以限制对您 Amazon S3 内容的访问，并为该

OAC 授予所需的权限。例如，如果因为您想使用 PUT 方法而将 CloudFront 配置为接受并转发这些方法，则必须配置 Amazon S3 存储桶策略来适当地处理 DELETE 请求，这样查看器就无法删除您不希望其删除的资源。有关更多信息，请参阅 the section called “限制对 Amazon Simple Storage Service 源的访问”。

有关 Amazon S3 所支持的操作的信息，请参阅 Amazon S3 文档。

CloudFront 删除或更新的 HTTP 请求标头

在将请求转发到 Amazon S3 源之前，CloudFront 将删除或更新一些标头。对于大多数标头，此行为与自定义源的相同。有关 HTTP 请求标头的完整列表以及 CloudFront 如何处理这些标头，请参阅HTTP 请求标头和 CloudFront 行为（自定义源和 Amazon S3 源）。

请求的最大长度与 URL 的最大长度

请求的最大长度，包括路径、查询字符串（如果有）以及标头，是 20480 个字节。

CloudFront 根据请求来构造 URL。此 URL 的最大长度是 8192 个字节。

如果请求或 URL 超出最大长度，则 CloudFront 将 HTTP 状态代码 413（请求实体过大）返回到查看器，然后终止与查看器的 TCP 连接。

OCSP Stapling

当查看器提交对象的 HTTPS 请求时，CloudFront 或查看器必须与证书颁发机构（CA，Certificate Authority）确认尚未吊销域的 SSL 证书。OCSP Stapling 允许 CloudFront 验证证书并缓存来自 CA 的响应，这将使客户端无需直接向 CA 验证证书，从而加快证书验证速度。

当 CloudFront 收到对同一域中对象的大量 HTTPS 请求时，OCSP Stapling 的性能改进会更明显。CloudFront 边缘站点中的每台服务器必须提交一个单独的验证请求。当 CloudFront 收到对同一个域的大量 HTTPS 请求时，边缘站点中的每台服务器很快将收到来自 CA 的响应，表示它可固定到 SSL 握手中的数据包。当查看器确认证书有效时，CloudFront 可以提供请求的对象。如果您的分配未在 CloudFront 边缘站点中产生大量流量，则新请求更有可能定向到尚未向 CA 验证证书的服务器。在这种情况下，查看器将单独执行验证步骤，并且 CloudFront 服务器将提供对象。该 CloudFront 服务器还会向 CA 提交验证请求，因此，在它下次收到包含同一域名的请求时，便已获得来自 CA 的验证响应。

协议

CloudFront 根据查看器请求的协议（HTTP 或 HTTPS），来向源服务器转发 HTTP 或 HTTPS 请求。

> **Important**
>
> 如果您的 Amazon S3 存储桶配置为网站终端节点，则您无法将 CloudFront 配置为使用 HTTPS 与您的源进行通信，因为 Amazon S3 不支持该配置中的 HTTPS 连接。

查询字符串

您可配置 CloudFront 是否将查询字符串参数转发到您的 Amazon S3 源。有关更多信息，请参阅 根据查询字符串参数缓存内容。

源连接超时和尝试次数

源连接超时 是 CloudFront 在尝试建立与源的连接时等待的秒数。

源连接尝试 是 CloudFront 尝试连接到源的次数。

这些设置共同决定了在故障转移到辅助源（对于源组）或向查看器返回错误响应之前，CloudFront 尝试连接源的时间。默认情况下，CloudFront 在尝试连接到辅助源或返回错误响应之前等待长达 30 秒（3 次尝试，每次 10 秒）。您可以通过指定更短的连接超时和/或更少的尝试次数来减少此时间。

有关更多信息，请参阅 控制源超时和尝试次数。

源响应超时

源响应超时（也称为源读取超时 或源请求超时）同时适用于以下两种情况：

- CloudFront 在将请求转发到源后等待响应的时间长度（以秒为单位）。
- CloudFront 从收到来自源的响应的一个数据包到收到下一个数据包之间等待的时间长度（以秒为单位）。

CloudFront 行为取决于查看器请求的 HTTP 方法：

- `GET` 和 `HEAD` 请求 – 如果源在 30 秒内未响应或停止响应达 30 秒，则 CloudFront 中断连接。如果指定的源连接尝试次数超过 1 次，CloudFront 将再次尝试获取完整响应。CloudFront 最多尝试 3 次，具体取决于源连接尝试次数 设置的值。如果在最终尝试期间，源未进行响应，则 CloudFront 将不会重试，直至它收到对同一个源上的内容的其他请求。
- `DELETE`、`OPTIONS`、`PATCH`、`PUT` 和 `POST` 请求 – 如果源在 30 秒内未做出响应，CloudFront 将中断连接并且不会再次尝试以联系源。如有必要，客户端可以重新提交请求。

您无法更改 Amazon S3 源（未 配置为静态网站托管的 S3 存储桶）的响应超时。

针对同一对象的并发请求（请求折叠）

如果 CloudFront 边缘站点收到对于某个对象的请求，而该对象不在缓存中或缓存的对象已过期，则 CloudFront 会立即将请求发送到源。但是，如果存在针对同一对象的并发请求，也就是说，如果在 CloudFront 收到对第一个请求的响应之前，对同一个对象（具有相同缓存密钥）的其他请求到达边缘站点，则 CloudFront 会在将其他请求转发到源之前暂停。这一短暂的暂停有助于减少源上的负载。CloudFront 会将来自原始请求的响应发送到它在暂停期间收到的所有请求。这称为请求折叠。在 CloudFront 日志中，第一个请求在 `x-edge-result-type` 字段中标识为 `Miss`，而折叠的请求标识为 `Hit`。有关 CloudFront 日志的更多信息，请参阅the section called “CloudFront 和边缘函数日志记录”。

CloudFront 只折叠共享缓存键 的请求。如果其他请求不共享相同的缓存键（例如，因为您将 CloudFront 配置为基于请求标头、Cookie 或查询字符串进行缓存），则 CloudFront 会将所有带唯一缓存键的请求转发到您的源。

如果您希望阻止所有请求折叠，则可使用托管式缓存策略 `CachingDisabled`，该策略还可阻止缓存。有关更多信息，请参阅 使用托管式缓存策略。

如果要阻止特定对象的请求折叠，您可以将缓存行为的最小 TTL 设置为 0，并配置源以发送 `Cache-Control: private`、`Cache-Control: no-store`、`Cache-Control: no-cache`、`Cache-Control: max-age=0` 或 `Cache-Control: s-maxage=0`。这些配置将会增加源上的负载，并为 CloudFront 等待对第一个请求的响应时暂停的并发请求带来额外的延迟。

> **Important**
>
> 目前，如果您在缓存策略、源请求策略或旧版缓存设置中启用 Cookie 转发，则 CloudFront 不支持请求折叠。

CloudFront 如何处理来自 Amazon S3 源的响应

了解 CloudFront 如何处理来自 Amazon S3 源的响应。

目录

- 最大可缓存文件大小
- 重新导向

已取消的请求

如果对象不在边缘缓存中，或者在 CloudFront 从您的源获取对象后，还未来得及传送请求的对象，查看器便终止了会话（例如关闭浏览器），则 CloudFront 不会将该对象缓存到边缘站点中。

CloudFront 删除或更新的 HTTP 响应标头

在将来自 Amazon S3 源的响应转发给查看器之前，CloudFront 将删除或更新以下标头字段：

- `X-Amz-Id-2`
- `X-Amz-Request-Id`
- `Set-Cookie` – 如果您将 CloudFront 配置为转发 Cookie，则它会将 `Set-Cookie` 标头字段转发给客户端。有关更多信息，请参阅 根据 Cookie 缓存内容。
- `Trailer`
- `Transfer-Encoding` – 如果您的 Amazon S3 源返回此标头字段，则在将响应返回给查看器之前，CloudFront 会将值设置为 `chunked`。
- `Upgrade`
- `Via` – 在对查看器的响应中，CloudFront 将值设置为以下内容：

  `Via: `*`http-version alphanumeric-string`*`.cloudfront.net (CloudFront)`

  例如，该值类似以下内容：

  ```
  Via: 1.1 1026589cc7887e7a0dc7827b4example.cloudfront.net (CloudFront)
  ```

最大可缓存文件大小

CloudFront 在其缓存中保存的响应正文的最大大小为 50 GB。这包括未指定 `Content-Length` 标头值的分块传输响应。

您可以使用范围请求将对象分为每个为 50 GB 或以下的段进行请求，从而使用 CloudFront 缓存大于此大小的对象。CloudFront 将会缓存这些段，因为每个段的大小都为 50 GB 或以下。查看器检索完对象的所有段后，可以重建更大的原始对象。有关更多信息，请参阅 使用范围请求缓存大型对象。

重新导向

您可配置 Amazon S3 存储桶以将所有请求重定向到另一个主机名称；这可能是另一个 Amazon S3 存储桶或 HTTP 服务器。如果您配置存储桶以重定向所有请求且该存储桶是 CloudFront 分配的源，建议您使用分配的域名（例如，d111111abcdef8.cloudfront.net）或与分配（例如，example.com）有关的备用域名 (CNAME) 将存储桶配置为将所有请求重定向到 CloudFront 分配。否则，查看器请求会绕过 CloudFront，直接从新源提供对象。

> **Note**
>
> 如果您重定向请求到备用域名，您还必须通过添加 CNAME 记录为您的域更新 DNS 服务。有关更多信息，请参阅 通过添加备用域名（CNAME）使用自定义 URL。

这是在您配置重定向所有请求的存储桶时所发生的：

1. 查看器（例如，浏览器）从 CloudFront 请求对象。
2. CloudFront 将请求转发到 Amazon S3 存储桶，而该存储桶是分配的源。
3. Amazon S3 返回 HTTP 状态码 301（永久移走）以及新位置。
4. CloudFront 缓存重定向的状态码和新位置，并将值返回给查看器。CloudFront 没有遵照重定向以从新位置获取对象。
5. 查看器发送对象的另一个请求，但查看器此次指定了其从 CloudFront 获取的新位置：
   - 如果 Amazon S3 存储桶使用分配的域名或备用域名将所有请求重定向到 CloudFront 分配，则 CloudFront 从新位置中的 Amazon S3 存储桶或 HTTP 服务器请求对象。当新位置返回对象时，CloudFront 将其返回给查看器并将其缓存在边缘站点。
   - 如果 Amazon S3 存储桶将请求重定向到另一个位置，第二个请求将绕过 CloudFront。新位置中的 Amazon S3 存储桶或 HTTP 服务器直接将对象返回给查看器，因此，对象从未在 CloudFront 边缘缓存中进行缓存。

自定义源的请求和响应行为

要了解在使用自定义源时 CloudFront 如何处理请求和响应，请参阅以下部分：

主题

- CloudFront 如何处理请求及如何将请求转发给您的自定义源
- CloudFront 如何处理自定义源的响应

CloudFront 如何处理请求及如何将请求转发给您的自定义源

了解 CloudFront 如何处理查看器请求以及如何将请求转发到您的自定义源。

目录

身份验证

如果您将 `Authorization` 标头转发给源，则可将源服务器配置为针对以下类型的请求而请求进行客户端身份验证：

- DELETE
- GET
- HEAD
- PATCH
- PUT
- POST

对于 OPTIONS 请求，仅 在您使用以下 CloudFront 设置时才可以配置客户端身份验证：

- 配置 CloudFront 以将 Authorization 标头转发到您的源
- 将 CloudFront 配置为不 缓存对 OPTIONS 请求的响应

有关更多信息，请参阅 配置 CloudFront 以转发 Authorization 标头。

您可以使用 HTTP 或 HTTPS 将请求转发到源服务器。有关更多信息，请参阅 将 HTTPS 与 CloudFront 结合使用。

缓存持续时间和最小 TTL

要控制对象在 CloudFront 缓存中保留多长时间后 CloudFront 便会将另一请求转发到您的源，您可以：

- 配置您的源，以添加 Cache-Control 或 Expires 标题字段到每个对象。
- 在 CloudFront 缓存行为中指定最短 TTL 的值。
- 使用默认值 24 小时。

有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

客户端 IP 地址

如果查看器将请求发送到 CloudFront 且不包含 X-Forwarded-For 请求标头，则 CloudFront 将从 TCP 连接获取查看器的 IP 地址，添加包含该 IP 地址的 X-Forwarded-For 标头，并将请求转发到源。例如，如果 CloudFront 从 TCP 连接获取的 IP 地址为 192.0.2.2，则它会将以下标头转发到源：

```
X-Forwarded-For: 192.0.2.2
```

如果查看器将请求发送到 CloudFront 且包含 X-Forwarded-For 请求标头，则 CloudFront 将从 TCP 连接获取查看器的 IP 地址，将该 IP 地址附加到 X-Forwarded-For 标头的末尾，并将请求转发到源。例如，如果查看器请求包含 X-Forwarded-For: 192.0.2.4,192.0.2.3，并且 CloudFront 从 TCP 连接获取的 IP 地址为 192.0.2.2，则它会将以下标头转发到源：

```
X-Forwarded-For: 192.0.2.4,192.0.2.3,192.0.2.2
```

一些应用程序（例如，负载均衡器（包括 Elastic Load Balancing）、Web 应用程序防火墙、反向代理、入侵防御系统以及 API Gateway）会将转发请求的 CloudFront 边缘服务器的 IP 地址附加到 X-Forwarded-For 标头的末尾。例如，如果 CloudFront 将 X-Forwarded-For: 192.0.2.2 包含在它转发给 ELB 的请求中，并且 CloudFront 边缘服务器的 IP 地址为 192.0.2.199，则 EC2 实例收到的请求将包含以下标头：

```
X-Forwarded-For: 192.0.2.2,192.0.2.199
```

> **Note**
>
> X-Forwarded-For 标头包含 IPv4 地址（例如 192.0.2.44）和 IPv6 地址（例如 2001:0db8:85a3::8a2e:0370:7334）。
> 另请注意，当前服务器 (CloudFront) 路径上的每个节点都可以修改 X-Forwarded-For 标头。有关更多信息，请参阅 RFC 7239 的第 8.1 节。也可以使用 CloudFront 边缘计算函数修改标头。

客户端 SSL 身份验证

CloudFront 不支持使用客户端 SSL 证书进行客户端身份验证。如果源要求客户端证书，则 CloudFront 将删除请求。

压缩

有关更多信息，请参阅 提供压缩文件。

有条件请求

当 CloudFront 从边缘缓存接收已过期对象的请求时，其将请求转发到源，以获取对象的最新版本或获取 CloudFront 边缘缓存已具有最新版本的源的确认。通常，当源最后发送对象到 CloudFront 时，其在响应中包括 ETag 值、LastModified 值或以上两种值。在 CloudFront 转发到源的新请求中，CloudFront 将添加以下一项或两项内容：

- If-Match 或 If-None-Match 标头，其包括对象过期版本的ETag值。

- `If-Modified-Since` 标头，其包含对象过期版本的`LastModified`值。

源使用此信息来确定对象是否已更新，以及是否将整个对象返回到 CloudFront 或只返回 HTTP 304 状态码（未修改）。

> **Note**
>
> 如果将 CloudFront 配置为转发 Cookie（全部或某个子集），则不支持 `If-Modified-Since` 和 `If-None-Match` 条件请求。
> 有关更多信息，请参阅 根据 Cookie 缓存内容。

Cookies

您可配置 CloudFront 以将 cookie 转发到您的源。有关更多信息，请参阅 根据 Cookie 缓存内容。

跨源资源共享 (CORS)

如果您希望 CloudFront 采用跨源资源共享设置，请配置 CloudFront 以将 `Origin` 标头转发到源。有关更多信息，请参阅 根据请求标头缓存内容。

加密

您可以要求查看器使用 HTTPS 将请求发送到 CloudFront，并要求 CloudFront 使用由查看器使用的协议将请求转发到您的自定义源。有关更多信息，请参阅以下分配设置：

- 查看器协议策略
- 协议（仅自定义源）

CloudFront 使用 SSLv3、TLSv1.0、TLSv1.1 和 TLSv1.2 协议将 HTTPS 请求转发给源服务器。对于自定义源，您可以选择您希望 CloudFront 在与源通信时使用的 SSL 协议：

- 如果您使用 CloudFront 控制台，请使用源 SSL 协议复选框选择协议。有关更多信息，请参阅 创建分配。
- 如果您使用的是 CloudFront API，请使用 `OriginSslProtocols` 元素来指定协议。有关更多信息，请参阅《Amazon CloudFront API 参考》中的 OriginSslProtocols 和 DistributionConfig。

如果源是 Amazon S3 存储桶，则 CloudFront 总是使用 TLSv1.2。

> **Important**
>
> 不支持其他版本的 SSL 和 TLS。

有关将 HTTPS 与 CloudFront 搭配使用的更多信息，请参阅将 HTTPS 与 CloudFront 结合使用。有关 CloudFront 支持的用于查看器和 CloudFront 之间以及 CloudFront 和您的源之间的 HTTPS 通信的密码的列表，请参阅查看器和 CloudFront 之间支持的协议和密码。

包含正文的 GET 请求

如果查看器 GET 请求包含正文，则 CloudFront 将 HTTP 状态代码 403（禁止）返回给查看器。

HTTP 方法

如果您将 CloudFront 配置为处理其支持的所有 HTTP 方法，则 CloudFront 会接受来自查看器的以下请求，并将这些请求转发给您的自定义源：

- DELETE
- GET
- HEAD
- OPTIONS
- PATCH
- POST
- PUT

CloudFront 始终缓存对 GET 和 HEAD 请求的响应。您也可以将 CloudFront 配置为缓存对 OPTIONS 请求的响应。CloudFront 不缓存对使用其他方法的请求的响应。

有关如何配置是否让自定义源处理这些方法的信息，请参阅该源的文档。

> **Important**
>
> 如果您将 CloudFront 配置为接受 CloudFront 支持的所有 HTTP 方法并将这些方法转发到您的源，请将您的源服务器配置为处理所有方法。例如，如果因为您想使用 POST 而将 CloudFront 配置为接受并转发这些方法，则必须将您的源服务器配置为适当处理 DELETE 请求，以便查看器无法删除您不希望其删除的资源。有关更多信息，请参阅您的 HTTP 服务器的文档。

HTTP 请求标头和 CloudFront 行为（自定义源和 Amazon S3 源）

下表列出了 HTTP 请求标头，您可以将其转发到自定义源和 Amazon S3 源（有例外情况需要注意）。对于每个标头，该表包含有关以下内容的信息：

- 如果您不将 CloudFront 配置为将标头转发给您的源（这会导致 CloudFront 基于标头值缓存您的对象），CloudFront 会做出怎样的行为。
- 您是否能将 CloudFront 配置为基于标头的标头值缓存对象。

  您可以将 CloudFront 配置为基于 `Date` 和 `User-Agent` 标头中的值缓存对象，但建议您不要这样做。这些标头具有许多可能的值，并且基于其值的缓存操作会导致 CloudFront 将更多请求转发到源。

有关基于标头值的缓存操作的更多信息，请参阅根据请求标头缓存内容。

标头	在您未将 CloudFront 配置为基于标头值进行缓存时的行为	支持基于标头值的缓存操作
其他定义的标头	旧缓存设置 – CloudFront 将标头转发到源。	是
`Accept`	CloudFront 删除标头。	是
`Accept-Charset`	CloudFront 删除标头。	是
`Accept-Encoding`	如果值包含 `gzip` 或 `br`，则 CloudFront 会将标准化的 `Accept-Encoding` 标头转发到源。 有关更多信息，请参阅 压缩支持和 提供压缩文件。	是
`Accept-Language`	CloudFront 删除标头。	是
`Authorization`	•	是

标头	在您未将 CloudFront 配置为基于标头值进行缓存时的行为	支持基于标头值的缓存操作
	GET 和 HEAD 请求 – 在将请求转发给您的源之前，CloudFront 将删除 Authorization 标头字段。 • OPTIONS 请求 – 如果您将 CloudFront 配置为缓存对 Authorization 请求的响应，则在将请求转发给您的源之前，CloudFront 将删除 OPTIONS 标头字段。 如果您未将 CloudFront 配置为缓存对 OPTIONS 请求的响应，则 CloudFront 会将 Authorization 标头字段转发给您的源。 • DELETE、PATCH、POST 和 PUT 请求 – 在将请求转发给您的源之前，CloudFront 不会删除标头字段。	
Cache-Control	CloudFront 将标头转发到源。	否
CloudFront-Forwarded-Proto	在将请求转发给您的源之前，CloudFront 不会添加标头。 有关更多信息，请参阅 配置基于请求协议的缓存。	是
CloudFront-Is-Desktop-Viewer	在将请求转发给您的源之前，CloudFront 不会添加标头。 有关更多信息，请参阅 配置基于设备类型的缓存。	是

标头	在您未将 CloudFront 配置为基于标头值进行缓存时的行为	支持基于标头值的缓存操作
`CloudFront-Is-Mobile-Viewer`	在将请求转发给您的源之前，CloudFront 不会添加标头。 有关更多信息，请参阅 配置基于设备类型的缓存。	是
`CloudFront-Is-Tablet-Viewer`	在将请求转发给您的源之前，CloudFront 不会添加标头。 有关更多信息，请参阅 配置基于设备类型的缓存。	是
`CloudFront-Viewer-Country`	在将请求转发给您的源之前，CloudFront 不会添加标头。	是
`Connection`	在将请求转发给您的源之前，CloudFront 会将此标头替换为 `Connection: Keep-Alive` 。	否
`Content-Length`	CloudFront 将标头转发到源。	否
`Content-MD5`	CloudFront 将标头转发到源。	是
`Content-Type`	CloudFront 将标头转发到源。	是
`Cookie`	如果您将 CloudFront 配置为转发 Cookie，则它会将 `Cookie` 标头字段转发给您的源。如果不这样做，CloudFront 会删除 `Cookie` 标头字段。有关更多信息，请参阅 根据 Cookie 缓存内容。	否
`Date`	CloudFront 将标头转发到源。	是，但建议不要这样做

标头	在您未将 CloudFront 配置为基于标头值进行缓存时的行为	支持基于标头值的缓存操作
Expect	CloudFront 删除标头。	是
From	CloudFront 将标头转发到源。	是
Host	CloudFront 将值设置为与请求的对象关联的源的域名。 您无法基于 Amazon S3 或 MediaStore 源的主机标头进行缓存。	是 (自定义) 否 (S3 和 MediaStore)
If-Match	CloudFront 将标头转发到源。	是
If-Modified-Since	CloudFront 将标头转发到源。	是
If-None-Match	CloudFront 将标头转发到源。	是
If-Range	CloudFront 将标头转发到源。	是
If-Unmodified-Since	CloudFront 将标头转发到源。	是
Max-Forwards	CloudFront 将标头转发到源。	否
Origin	CloudFront 将标头转发到源。	是
Pragma	CloudFront 将标头转发到源。	否
Proxy-Authenticate	CloudFront 删除标头。	否

标头	在您未将 CloudFront 配置为基于标头值进行缓存时的行为	支持基于标头值的缓存操作
Proxy-Authorization	CloudFront 删除标头。	否
Proxy-Connection	CloudFront 删除标头。	否
Range	CloudFront 将标头转发到源。有关更多信息，请参阅 CloudFront 如何处理对象的部分请求 (Range GET)。。	是 (默认值)
Referer	CloudFront 删除标头。	是
Request-Range	CloudFront 将标头转发到源。	否
TE	CloudFront 删除标头。	否
Trailer	CloudFront 删除标头。	否
Transfer-Encoding	CloudFront 将标头转发到源。	否
Upgrade	CloudFront 会删除标头，除非您建立了 WebSocket 连接。	否（WebSocket 连接除外）
User-Agent	CloudFront 将该标头字段的值替换为 Amazon CloudFront 。如果您希望 CloudFront 根据用户使用的设备缓存您的内容，请参阅配置基于设备类型的缓存。	是，但建议不要这样做

标头	在您未将 CloudFront 配置为基于标头值进行缓存时的行为	支持基于标头值的缓存操作
`Via`	CloudFront 将标头转发到源。	是
`Warning`	CloudFront 将标头转发到源。	是
`X-Amz-Cf-Id`	在将请求转发给您的源之前，CloudFront 会将标头添加到查看器请求。标头值包含一个用于唯一标识请求的加密的字符串。	否
`X-Edge-*`	CloudFront 删除所有 `X-Edge-*` 标头。	否
`X-Forwarded-For`	CloudFront 将标头转发到源。有关更多信息，请参阅 客户端 IP 地址。	是
`X-Forwarded-Proto`	CloudFront 删除标头。	否
`X-HTTP-Method-Override`	CloudFront 删除标头。	是
`X-Real-IP`	CloudFront 删除标头。	否

HTTP 版本

CloudFront 使用 HTTP/1.1 将请求转发给您的自定义源。

请求的最大长度与 URL 的最大长度

请求的最大长度，包括路径、查询字符串（如果有）以及标头，是 20480 个字节。

CloudFront 根据请求来构造 URL。此 URL 的最大长度是 8192 个字节。

如果请求或 URL 超出这些最大值，则 CloudFront 将 HTTP 状态代码 413（请求实体过大）返回到查看器，然后终止与查看器的 TCP 连接。

OCSP Stapling

当查看器提交对象的 HTTPS 请求时，CloudFront 或查看器必须与证书颁发机构 (CA) 确认尚未吊销域的 SSL 证书。OCSP Stapling 允许 CloudFront 验证证书并缓存来自 CA 的响应，这将使客户端无需直接向 CA 验证证书，从而加快证书验证速度。

当 CloudFront 收到对同一域中对象的大量 HTTPS 请求时，OCSP Stapling 的性能改进会更明显。CloudFront 边缘站点中的每台服务器必须提交一个单独的验证请求。当 CloudFront 收到同一域的大量 HTTPS 请求时，边缘站点中的每台服务器很快将收到来自 CA 的响应，表示它可“固定”到 SSL 握手中的数据包；当查看器对证书有效感到满意时，CloudFront 可提供请求的对象。如果您的分配未在 CloudFront 边缘站点中产生大量流量，则新请求更有可能定向到尚未向 CA 验证证书的服务器。在这种情况下，查看器将单独执行验证步骤，并且 CloudFront 服务器将提供对象。该 CloudFront 服务器还会向 CA 提交验证请求，因此，在它下次收到包含同一域名的请求时，便已获得来自 CA 的验证响应。

持久性连接

当 CloudFront 收到来自您的源的响应时，它会尝试将连接保持几秒钟，以防在此时段内有另一个请求到达。保持持久性连接可节省重新建立 TCP 连接以及针对后续请求再次进行 TLS 握手所需的时间。

有关更多信息 (包括如何配置持久性连接)，请参阅源保持连接超时（仅自定义源）一节中的分配设置参考。

协议

CloudFront 基于以下将 HTTP 或 HTTPS 请求转发到源服务器：

- 查看器发送到 CloudFront 的请求的协议（HTTP 或 HTTPS）。
- CloudFront 控制台中源协议策略的值，或者，如果您使用 CloudFront API，则为 `OriginProtocolPolicy`复杂型中的 `DistributionConfig` 元素。在 CloudFront 控制台中，选项包括仅 HTTP、仅 HTTPS 和匹配查看器。

如果您指定仅 HTTP 或仅 HTTPS，则 CloudFront 仅使用指定的协议将请求转发到源服务器，而不考虑查看器请求中的协议。

如果您指定匹配查看器，则 CloudFront 将使用查看器请求中的协议将请求转发到源服务器。请注意，CloudFront 仅缓存对象一次，即使查看器使用 HTTP 和 HTTPS 协议发出请求。

> **Important**
>
> 如果 CloudFront 使用 HTTPS 协议将请求转发给源，并且源服务器返回无效证书或自签名证书，则 CloudFront 将中断 TCP 连接。

有关使用 CloudFront 控制台如何更新分配的信息，请参阅更新分配。有关如何使用 CloudFront API 更新分配的信息，请转到《Amazon CloudFront API 参考》中的 UpdateDistribution。

查询字符串

您可配置 CloudFront 是否将查询字符串参数转发到您的源。有关更多信息，请参阅 根据查询字符串参数缓存内容。

源连接超时和尝试次数

源连接超时 是 CloudFront 在尝试建立与源的连接时等待的秒数。

源连接尝试 是 CloudFront 尝试连接到源的次数。

这些设置共同决定了在故障转移到辅助源（对于源组）或向查看器返回错误响应之前，CloudFront 尝试连接源的时间。默认情况下，CloudFront 在尝试连接到辅助源或返回错误响应之前等待长达 30 秒（3 次尝试，每次 10 秒）。您可以通过指定更短的连接超时和/或更少的尝试次数来减少此时间。

有关更多信息，请参阅 控制源超时和尝试次数。

源响应超时

源响应超时（也称为源读取超时 或源请求超时）同时适用于以下两种情况：

- CloudFront 在将请求转发到源后等待响应的时间长度（以秒为单位）。
- CloudFront 从收到来自源的响应的一个数据包到收到下一个数据包之间等待的时间长度（以秒为单位）。

CloudFront 行为取决于查看器请求的 HTTP 方法：

- `GET` 和 `HEAD` 请求 – 如果源在响应超时的持续时间内没有响应或停止响应，则 CloudFront 中断连接。如果指定的源连接尝试次数超过 1 次，CloudFront 将再次尝试获取完整响应。CloudFront 最多尝试 3 次，具体取决于源连接尝试次数 设置的值。如果在最终尝试期间，源未进行响应，则 CloudFront 将不会重试，直至它收到对同一个源上的内容的其他请求。

- DELETE、OPTIONS、PATCH、PUT 和 POST 请求 – 如果源在 30 秒内未做出响应，CloudFront 将中断连接并且不会再次尝试以联系源。如有必要，客户端可以重新提交请求。

有关更多信息（包括如何配置源响应超时），请参阅响应超时（仅自定义源）。

针对同一对象的并发请求（请求折叠）

如果 CloudFront 边缘站点收到对于某个对象的请求，而该对象不在缓存中或缓存的对象已过期，则 CloudFront 会立即将请求发送到源。但是，如果存在针对同一对象的并发请求，也就是说，如果在 CloudFront 收到对第一个请求的响应之前，对同一个对象（具有相同缓存密钥）的其他请求到达边缘站点，则 CloudFront 会在将其他请求转发到源之前暂停。这一短暂的暂停有助于减少源上的负载。CloudFront 会将来自原始请求的响应发送到它在暂停期间收到的所有请求。这称为请求折叠。在 CloudFront 日志中，第一个请求在 `x-edge-result-type` 字段中标识为 `Miss`，而折叠的请求标识为 `Hit`。有关 CloudFront 日志的更多信息，请参阅the section called “CloudFront 和边缘函数日志记录”。

CloudFront 只折叠共享缓存键 的请求。如果其他请求不共享相同的缓存键（例如，因为您将 CloudFront 配置为基于请求标头、Cookie 或查询字符串进行缓存），则 CloudFront 会将所有带唯一缓存键的请求转发到您的源。

如果您希望阻止所有请求折叠，则可使用托管式缓存策略 `CachingDisabled`，该策略还可阻止缓存。有关更多信息，请参阅 使用托管式缓存策略。

如果要阻止特定对象的请求折叠，您可以将缓存行为的最小 TTL 设置为 0，并配置源以发送 `Cache-Control: private`、`Cache-Control: no-store`、`Cache-Control: no-cache`、`Cache-Control: max-age=0` 或 `Cache-Control: s-maxage=0`。这些配置将会增加源上的负载，并为 CloudFront 等待对第一个请求的响应时暂停的并发请求带来额外的延迟。

> **Important**
>
> 目前，如果您在缓存策略、源请求策略或旧版缓存设置中启用 Cookie 转发，则 CloudFront 不支持请求折叠。

`User-Agent` 标头

如果您希望 CloudFront 基于用户用来查看内容的设备缓存不同版本的对象，建议您将 CloudFront 配置为将一个或多个以下标头转发给您的自定义源：

- CloudFront-Is-Desktop-Viewer
- CloudFront-Is-Mobile-Viewer
- CloudFront-Is-SmartTV-Viewer
- CloudFront-Is-Tablet-Viewer

根据 User-Agent 标头的值，在将请求转发给您的源之前，CloudFront 将这些标头的值设置为 true 或 false。如果某个设备归入多个类别中，则多个值可能为 true。例如，对于一些平板电脑设备，CloudFront 可能将 CloudFront-Is-Mobile-Viewer 和 CloudFront-Is-Tablet-Viewer 设置为 true。有关将 CloudFront 配置为基于请求标头进行缓存的更多信息，请参阅根据请求标头缓存内容。

您可以将 CloudFront 配置为基于 User-Agent 标头中的值缓存对象，但建议您不要这样做。User-Agent 标头具有许多可能的值，并且根据这些值的缓存操作导致 CloudFront 将更多请求转发到源。

如果未将 CloudFront 配置为根据 User-Agent 标头中的值缓存对象，在将请求转发到源之前，CloudFront 将添加具有以下值的 User-Agent 标头：

```
User-Agent = Amazon CloudFront
```

无论来自查看器的请求是否包含 User-Agent 标头，CloudFront 都会添加此标头。如果来自查看器的请求包含 User-Agent 标头，则 CloudFront 会删除它。

CloudFront 如何处理自定义源的响应

了解 CloudFront 如何处理自定义源的响应。

目录

- 重新导向
- Transfer-Encoding 标头

`100 Continue` 响应

您的源服务器不能向 CloudFront 发送多个 100-Continue 响应。在第一个 100-Continue 响应之后，CloudFront 需要“HTTP 200 正常”响应。如果您的源服务器在第一个 100-Continue 响应之后又发送了该响应，CloudFront 将返回错误。

缓存

- 确保源服务器为 `Date` 和 `Last-Modified` 标头字段设置有效、准确的值。
- CloudFront 通常会遵循来自源的响应中的 `Cache-Control: no-cache` 标头。有关例外，请参阅针对同一对象的并发请求（请求折叠）。

已取消的请求

如果对象不在边缘缓存中，或者在 CloudFront 从您的源获取对象后，还未来得及传送请求的对象，查看器便终止了会话（例如关闭浏览器），则 CloudFront 不会将该对象缓存到边缘站点中。

内容协商

如果您的源在响应中返回 `Vary:*`，并且相应缓存行为的最短 TTL 的值为 0，则 CloudFront 将缓存对象，但仍会将对象的每个后续请求转发到源以确认缓存包含最新版本的对象。CloudFront 不包含任何条件标头，例如 `If-None-Match` 或 `If-Modified-Since`。因此，您的源会将对象返回给 CloudFront 以响应每个请求。

如果您的源在响应中返回 `Vary:*`，并且相应缓存行为的最短 TTL 的值为任何其他值，则 CloudFront 将处理 `Vary` 标头，如CloudFront 移除或替换的 HTTP 响应标头中所述。

Cookie

如果您为缓存行为启用了 cookie，并且如果源返回带对象的 cookie，CloudFront 则缓存对象和 cookie。请注意，这降低了对象的缓存能力。有关更多信息，请参阅 根据 Cookie 缓存内容。

中断的 TCP 连接

如果在您的源将对象返回给 CloudFront 时，CloudFront 和您的源之间的 TCP 连接中断，CloudFront 行为将取决于您的源是否在响应中包括 `Content-Length` 标头：

- Content-Length 标头 – CloudFront 在其从您的源中获得对象时将对象返回给查看器。但是，如果 `Content-Length` 标头的值与对象大小不匹配，CloudFront 则不缓存对象。
- Transfer-Encoding: 分块 – 在从源中获取对象时，CloudFront 将对象返回到查看器。但是，如果分块响应未完成，则 CloudFront 不会对该对象进行缓存。
- 无 Content-Length 标头 – CloudFront 将对象返回给查看器并对其进行缓存，但该对象可能不完整。在没有 `Content-Length` 标头的情况下，CloudFront 无法确定 TCP 连接是否是偶然中断还是有意中断。

建议您配置您的 HTTP 服务器，以添加 `Content-Length` 标头，防止 CloudFront 缓存部分对象。

CloudFront 移除或替换的 HTTP 响应标头

在将来自您的源的响应转发给查看器之前，CloudFront 将删除或更新以下标头字段：

- `Set-Cookie` – 如果您将 CloudFront 配置为转发 Cookie，则它会将 `Set-Cookie` 标头字段转发给客户端。有关更多信息，请参阅 根据 Cookie 缓存内容。
- `Trailer`
- `Transfer-Encoding` – 如果您的源返回此标头字段，则在将响应返回给查看器之前，CloudFront 会将值设置为 `chunked`。
- `Upgrade`
- `Vary` – 请注意以下几点：
  - 如果您将 CloudFront 配置为将任何设备特定的标头转发给您的源 (`CloudFront-Is-Desktop-Viewer`、`CloudFront-Is-Mobile-Viewer`、`CloudFront-Is-SmartTV-Viewer`、`CloudFront-Is-Tablet-Viewer`)，并且您将源配置为将 `Vary:User-Agent` 返回给 CloudFront，则 CloudFront 会将 `Vary:User-Agent` 返回给查看器。有关更多信息，请参阅 配置基于设备类型的缓存。
  - 如果您将源配置为将 `Accept-Encoding` 或 `Cookie` 包含在 `Vary` 标头中，则 CloudFront 会将这些值包含在对查看器的响应中。
  - 如果配置 CloudFront 以将标头转发到源，并且配置源以通过 `Vary` 标头将标头名返回到 CloudFront（如 `Vary:Accept-Charset,Accept-Language`），则 CloudFront 将具有这些值的 `Vary` 标头返回到查看器。
  - 有关 CloudFront 如何处理 * 标头中的 `Vary` 值的信息，请参阅内容协商。
  - 如果您将源配置为将任何其他值包含在 `Vary` 标头中，则在将响应返回给查看器之前，CloudFront 将删除这些值。
- `Via` – 在对查看器的响应中，CloudFront 将值设置为以下内容：

```
Via: http-version alphanumeric-string.cloudfront.net (CloudFront)
```

例如，该值类似以下内容：

```
Via: 1.1 1026589cc7887e7a0dc7827b4example.cloudfront.net (CloudFront)
```

最大可缓存文件大小

CloudFront 在其缓存中保存的响应正文的最大大小为 50 GB。这包括未指定 `Content-Length` 标头值的分块传输响应。

您可以使用范围请求将对象分为每个为 50 GB 或以下的段进行请求，从而使用 CloudFront 缓存大于此大小的对象。CloudFront 将会缓存这些段，因为每个段的大小都为 50 GB 或以下。查看器检索完对象的所有段后，可以重建更大的原始对象。有关更多信息，请参阅 使用范围请求缓存大型对象。

源不可用

如果您的源服务器不可用且 CloudFront 收到针对边缘缓存中已过期对象的请求（例如，因为 `Cache-Control max-age` 指令中指定的期限已过），那么 CloudFront 会提供该对象的已过期版本或提供自定义错误页面。有关配置自定义错误页面时 CloudFront 行为的更多信息，请参阅当您已配置自定义错误页面时 CloudFront 如何处理错误。

某些情况下，将逐出很少被请求的对象且不再在边缘缓存中提供。CloudFront 不能提供已被逐出的对象。

重新导向

如果更改对象在源服务器上的位置，您可以配置 Web 服务器以重定向请求到新位置。在您配置重定向后，查看器第一次提交对象请求时，CloudFront 将请求发送到源，源则响应重定向（例如 `302 Moved Temporarily`）。CloudFront 缓存重定向并将其返回给查看器。CloudFront 不遵照重定向。

您可以配置 Web 服务器以将请求重定向到以下位置之一：

- 对象在原服务器上的新 URL。当查看器按照重定向路线访问新 URL 时，查看器将绕过 CloudFront，直接到达源。因此，建议您不要将请求重定向到对象在源上的新 URL。
- 对象的新 CloudFront URL。当查看器提交包含新 CloudFront URL 的请求时，CloudFront 将从源上的新位置获取对象，并将其缓存在边缘站点中，然后将该对象返回给查看器。对象随后的请求将由边缘站点提供。这避免了查看器从源请求对象相关的延迟和负载。但是，对象的每个新请求将产生两个 CloudFront 请求的费用。

`Transfer-Encoding` 标头

CloudFront 仅支持 chunked 标头的 `Transfer-Encoding` 值。如果您的源返回 `Transfer-Encoding: chunked`，则 CloudFront 会在边缘站点收到对象后将该对象返回给客户端，然后以分块格式将该对象缓存起来以提供给后续请求。

如果查看器发出 `Range GET` 请求，并且源返回 `Transfer-Encoding: chunked`，则 CloudFront 会将整个对象返回给查看器而不是请求的范围。

如果无法预先确定响应的内容长度，建议您使用分块编码。有关更多信息，请参阅 中断的 TCP 连接。

源组的请求和响应行为

对源组的请求与对未设置为源组的源的请求相同，但存在源故障转移时除外。与任何其他源一样，当 CloudFront 收到请求且内容已在边缘站点中缓存时，内容将通过缓存提供给查看器。当存在缓存未命中且源是源组时，查看器请求将转发到源组中的主要源。

主源的请求和响应行为与未在源组中的源相同。有关更多信息，请参阅 Amazon S3 源的请求和响应行为和 自定义源的请求和响应行为。

下面描述了当主源返回特定 HTTP 状态代码时源故障转移的行为：

- HTTP 2xx 状态代码（成功）：CloudFront 将缓存文件并将它返回到查看器。
- HTTP 3xx 状态代码（重定向）：CloudFront 将状态代码返回到查看器。
- HTTP 4xx 或 5xx 状态代码（客户端/服务器错误）：如果已为故障转移配置返回的状态代码，则 CloudFront 将相同请求发送到源组中的次要源。
- HTTP 4xx 或 5xx 状态代码（客户端/服务器错误）：如果尚未为故障转移配置返回的状态代码，则 CloudFront 会将错误返回到查看器。

只有当查看器请求的 HTTP 方法是 `GET`、`HEAD` 或 `OPTIONS` 时，CloudFront 才会故障转移到辅助源。当查看器发送不同的 HTTP 方法（例如 `POST`、`PUT` 等）时，CloudFront 不会进行故障转移。

当 CloudFront 向辅助源发送请求时，响应行为与不在源组中的 CloudFront 源的响应行为相同。

有关源组的更多信息，请参阅通过 CloudFront 源失效转移来优化高可用性。

向源请求添加自定义标头

您可以配置 CloudFront 以便向发送到您的源的请求中添加自定义标头。您可以使用这些自定义标头，从源发送和收集通过典型查看器请求无法获得的信息。您甚至可以针对每个源自定义这些标头。CloudFront 支持用于自定义源和 Amazon S3 源的自定义标头。

目录

使用案例

您可以使用自定义标头，如下例所示：

识别来自 CloudFront 的请求

您可以标识您的源从 CloudFront 接收的请求。如果您想知道用户是否绕过 CloudFront，或者您使用了多个 CDN 并且希望了解有关哪些请求来自每个 CDN 的信息，此功能很有用。

> **Note**
>
> （如果您使用 Amazon S3 源并启用 Amazon S3 服务器访问日志记录，日志将不包含标头信息。）

确定哪些请求来自于特定分配

如果您配置多个 CloudFront 分配以使用相同的源，则可以在每个分配中添加不同的自定义标头。然后，可以使用来自您的源中的日志，确定哪些请求来自于哪个 CloudFront 分配。

允许跨源资源共享 (CORS)

如果您的某些查看器不支持跨源资源共享 (CORS)，则可以配置 CloudFront，以便始终将 `Origin` 标头添加到发送到您的源的请求中。然后，可以配置您的源，以便为每个请求返回 `Access-Control-Allow-Origin` 标头。您还必须将 CloudFront 配置为遵守 CORS 设置。

控制对内容的访问

可以使用自定义标头来控制对内容的访问。通过配置源以便仅在请求包含 CloudFront 添加的自定义标头时才响应请求，您可以防止用户绕过 CloudFront 并直接访问源上的内容。有关更多信息，请参阅 在自定义源上限制对文件的访问。

配置 CloudFront 以便向源请求添加自定义标头

要配置分配以便向其发送到源的请求添加自定义标头，请使用以下方法之一更新源配置：

- CloudFront 控制台 – 在创建或更新分配时，请在添加自定义标头设置中指定标头名称和值。有关更多信息，请参阅 添加自定义标头。
- CloudFront API – 对于要添加自定义标头的每个源，请在 `Origin` 内部的 `CustomHeaders` 字段中指定标头名称和值。有关更多信息，请参阅《Amazon CloudFront API 参考》中的 CreateDistribution 或 UpdateDistribution。

如果您指定的标头名称和值在查看器请求中尚不存在，CloudFront 会将这些标头名称和值添加到源请求。如果标头存在，CloudFront 会在将请求转发到源之前覆盖该标头值。

有关适用于源自定义标头的配额，请参阅标头的配额。

CloudFront 无法添加到源请求的自定义标头

您无法配置 CloudFront 以便将以下任意标头添加到发送到您的源的请求：

- `Cache-Control`
- `Connection`
- `Content-Length`
- `Cookie`
- `Host`
- `If-Match`
- `If-Modified-Since`
- `If-None-Match`
- `If-Range`
- `If-Unmodified-Since`

- Max-Forwards
- Pragma
- Proxy-Authorization
- Proxy-Connection
- Range
- Request-Range
- TE
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- 以 X-Amz- 开头的标头
- 以 X-Edge- 开头的标头
- X-Real-Ip

配置 CloudFront 以转发 **Authorization** 标头

当 CloudFront 转发查看器请求到您的源时，默认情况下，CloudFront 会删除一些查看器标头，包括 Authorization 标头。为了确保您的源始终能接收到源请求中的 Authorization 标头，您可以选择以下几种方式：

- 使用缓存策略将 Authorization 标头添加到缓存键中。源请求会自动包含缓存键中的所有标头。有关更多信息，请参阅 使用策略来控制缓存键。
- 使用将所有查看器标头转发到源的源请求策略。您无法通过源请求策略单独转发 Authorization 标头；当您使用该策略转发所有查看器标头时，CloudFront 会将 Authorization 标头包含在查看器请求中。针对此用例，CloudFront 提供了名为 Managed-AllViewer 的托管源请求策略。有关更多信息，请参阅 使用托管式源请求策略。

CloudFront 如何处理对象的部分请求 (Range GET)。

对于大型对象，查看器（Web 浏览器或客户端）可能做出多个 GET 请求并使用 Range 请求标头以较小的段下载对象。字节范围的这些请求，有时也被称为 Range GET 请求，提高了部分下载的效率和恢复部分失败的传输。

当 CloudFront 接收 Range GET 请求时，其检查接收请求的边缘站点中的缓存。如果边缘站点中的缓存已经包含整个对象或请求的对象段，CloudFront 会立即从缓存提供所请求的范围。

如果缓存不包含所请求的范围，CloudFront 会将请求转发到源。（要优化性能，CloudFront 可请求比客户端在 Range GET 中请求的范围更大的范围。）接下来会发生的操作取决于源是否支持 Range GET 请求：

- 源支持 **Range GET** 请求时 – 返回所请求的范围。CloudFront 提供被请求的范围，而且还对其进行缓存以用于将来的请求。（与许多 HTTP 服务器一样，Amazon S3 支持 Range GET 请求。）
- 源不支持 **Range GET** 请求时 – 返回整个对象。CloudFront 通过发送整个对象来满足当前请求，同时缓存它以用于将来的请求。在 CloudFront 在边缘缓存中缓存整个对象后，它通过提供被请求的范围响应新的 Range GET 请求。

无论是哪种情况，CloudFront 都会在第一个字节到达后从源开始向最终用户提供所请求的范围或对象。

> **Note**
>
> 如果查看器发出 Range GET 请求，并且源返回 `Transfer-Encoding: chunked`，则 CloudFront 会将整个对象返回给查看器而不是请求的范围。

CloudFront 一般遵照 Range 标头的 RFC 规范。但是，如果您的 Range 标头不遵守以下要求，CloudFront 将返回 HTTP 状态码 200 与完整的对象，而不是状态码 206 与指定的范围：

- 范围必须列按升序排列。例如，`100-200,300-400` 是有效的，`300-400,100-200` 是无效的。
- 范围不能重叠。例如，`100-200,150-250` 是无效的。
- 所有范围的规范必须有效。例如，您不能指定负值作为范围的一部分。

有关 Range 请求标头的更多信息，请参阅 RFC 7233 中的范围请求，或者 MDN Web Docs 中的范围。

使用范围请求缓存大型对象

启用缓存后，CloudFront 不会检索或缓存大于 50 GB 的对象。如果源表示对象大于此大小（在 `Content-Length` 响应标头中），CloudFront 将关闭与源的连接并向查看器返回一条错误。（禁用缓

存后，CloudFront 可以从源检索大于此大小的对象，然后将其传递给查看器。但是，CloudFront 不会缓存该对象。）

但借助范围请求，您可以使用 CloudFront 缓存超过最大可缓存文件大小的对象。

Example 示例

1. 假设某个源中具有一个 100 GB 的对象。启用缓存后，CloudFront 不会检索或缓存这样大的对象。但查看器可以发送多个范围请求以分段检索此对象，每个段小于 50 GB。
2. 查看器可以发送一个带有 `Range: bytes=0-21474836480` 标头的请求检索第一段，发送另一个带有 `Range: bytes=21474836481-42949672960` 的请求以检索下一段，依此类推，从而按 20 GB 的段请求该对象。
3. 查看器收到所有段后，可以将这些段组合起来构建原始的 100GB 对象。
4. 在此例中，CloudFront 会缓存该对象的所有 20GB 段，并且可以响应对缓存中相同段的后续请求。

CloudFront 如何处理来自源的 HTTP 3xx 状态代码

当 CloudFront 从 Amazon S3 存储桶或自定义源服务器请求对象时，您的源有时会返回 HTTP 3xx 状态代码。这通常表明出现下列问题之一：

- 对象的 URL 已更改（例如，状态代码 301、302、307 或 308）
- 自 CloudFront 上次请求该对象以来，该对象未发生更改（状态代码 304）

CloudFront 根据 CloudFront 分配中的设置和响应中的标头缓存 3xx 响应。仅当您在响应中包含来自源的 `Cache-Control` 标头时，CloudFront 才会缓存 307 和 308 响应。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

如果您的源返回重定向状态代码（例如 301 或 307），则 CloudFront 不遵循重定向操作。CloudFront 将 301 或 307 响应传递给查看器，查看器可以通过发送新请求来遵循重定向操作。

CloudFront 如何处理来自源的 HTTP 4xx 和 5xx 状态代码

当 CloudFront 从 Amazon S3 存储桶或自定义源服务器中请求对象时，源有时会返回 HTTP 4xx 或 5xx 状态代码，这表示出现了错误。CloudFront 的行为取决于：

- 您是否已配置自定义错误页面
- 您是否已配置希望 CloudFront 缓存来自源的错误响应的时间长度（错误缓存最小 TTL）

- 状态代码
- 对于 5xx 状态代码，请求的对象当前是否在 CloudFront 边缘缓存中
- 对于一些 4xx 状态代码，源返回 `Cache-Control max-age` 还是 `Cache-Control s-maxage` 标头

CloudFront 始终缓存对 `GET` 和 `HEAD` 请求的响应。您也可以将 CloudFront 配置为缓存对 `OPTIONS` 请求的响应。CloudFront 不缓存对使用其他方法的请求的响应。

如果源无响应，则对源的 CloudFront 请求超时，这被认为是来自源的 HTTP 5xx 错误，即使源没有使用该错误做出响应也是如此。在这种情况下，CloudFront 继续处理缓存内容。有关更多信息，请参阅源不可用。

如果您已启用日志记录，无论 HTTP 状态代码是什么，CloudFront 均会将结果写入日志中。

有关与 CloudFront 返回的错误消息相关的功能和选项的更多信息，请参阅以下内容：

- 有关在 CloudFront 控制台中设置自定义错误页面的信息，请参阅自定义错误页面和错误缓存。
- 有关 CloudFront 控制台中的错误缓存最小 TTL 的信息，请参阅错误缓存最小 TTL（秒）。
- 有关 CloudFront 缓存的 HTTP 状态代码列表，请参阅CloudFront 缓存的 HTTP 4xx 和 5xx 状态代码。

主题

当您已配置自定义错误页面时 CloudFront 如何处理错误

如果您已配置自定义错误页面，则 CloudFront 的行为取决于请求的对象是否在边缘缓存中。

请求的对象未在边缘缓存中

在满足以下所有条件时，CloudFront 将继续尝试从您的源获取请求的对象：

- 查看器请求对象。
- 对象不在边缘缓存中。
- 您的源返回 HTTP 4xx 或 5xx 状态代码并且出现下列情况之一：

- 源返回 HTTP 5xx 状态代码，而不是返回 304 状态代码（未修改）或更新版本的对象。
- 您的源返回不受缓存控制标头限制的 HTTP 4xx 状态代码，并且包括在以下状态代码列表中：CloudFront 始终缓存的 HTTP 4xx 和 5xx 状态代码
- 您的源返回没有 Cache-Control max-age 标头或 Cache-Control s-maxage 标头的 HTTP 4xx 状态代码，并且包括在以下状态代码列表中：控制 CloudFront 根据 Cache-Control 标头缓存的 HTTP 4xx 的状态代码。

CloudFront 将执行以下操作：

1. 在收到查看器请求的 CloudFront 边缘缓存中，CloudFront 检查您的分配配置，并获取与源返回的状态代码对应的自定义错误页面的路径。
2. CloudFront 在您的分配中查找路径模式与自定义错误页面的路径相匹配的第一项缓存行为。
3. CloudFront 边缘站点会将针对自定义错误页面的请求发送到在缓存行为中指定的源。
4. 源将自定义错误页面返回到边缘站点。
5. CloudFront 将自定义错误页面返回给发出请求的查看方，还会将自定义错误页面缓存以下时间中的最大值：
   - 错误缓存最小 TTL 指定的时间量（默认情况下为 10 秒）
   - 在第一个请求生成错误时，源返回的 Cache-Control max-age 标头或 Cache-Control s-maxage 标头指定的时间量
6. 经过了缓存时间（在第 5 步中确定）之后，CloudFront 向源转发另一个请求以再次尝试获取请求的对象。CloudFront 将继续按照由最短错误缓存 TTL 指定的间隔重试。

请求的对象在边缘缓存中

在满足以下所有条件时，CloudFront 将继续提供当前在边缘缓存中的对象：

- 查看器请求对象。
- 对象在边缘缓存中但已过期。
- 源返回 HTTP 5xx 状态代码，而不是返回 304 状态代码（未修改）或更新版本的对象。

CloudFront 将执行以下操作：

1. 如果您的源返回 5xx 状态代码，则 CloudFront 将提供对象，即使它已过期。在错误缓存最小 TTL 的持续时间内，CloudFront 会通过从边缘缓存提供对象来继续响应查看器请求。

如果您的源返回一个 4xx 状态代码，则 CloudFront 会将该状态代码而不是请求的对象返回给查看器。

2. 在最短错误缓存 TTL 过后，CloudFront 会通过再向您的源转发一个请求来再次尝试获取请求的对象。请注意，如果未频繁请求该对象，当您的源服务器仍返回 5xx 响应时，CloudFront 可能会将该对象从边缘缓存中逐出。有关对象在 CloudFront 边缘缓存中的保留时长的信息，请参阅管理内容保留在缓存中的时间长度（过期）。

当您尚未配置自定义错误页面时 CloudFront 如何处理错误

如果您尚未配置自定义错误页面，则 CloudFront 的行为取决于请求的对象是否在边缘缓存中。

请求的对象未在边缘缓存中

在满足以下所有条件时，CloudFront 将继续尝试从您的源获取请求的对象：

- 查看器请求对象。
- 对象不在边缘缓存中。
- 您的源返回 HTTP 4xx 或 5xx 状态代码并且出现下列情况之一：
  - 源返回 HTTP 5xx 状态代码，而不是返回 304 状态代码（未修改）或更新版本的对象。
  - 您的源返回不受缓存控制标头限制的 HTTP 4xx 状态代码，并且包括在以下状态代码列表中：CloudFront 始终缓存的 HTTP 4xx 和 5xx 状态代码
  - 您的源返回没有 `Cache-Control max-age` 标头或 `Cache-Control s-maxage` 标头的 HTTP 4xx 状态代码，并且包括在以下状态代码列表中：控制 CloudFront 根据 `Cache-Control` 标头缓存的 HTTP 4xx 的状态代码。

CloudFront 将执行以下操作：

1. CloudFront 将 4xx 或 5xx 状态代码返回给查看器，同时将状态代码在接收请求的边缘缓存中缓以下时间中的最大值：
   - 错误缓存最小 TTL 指定的时间量（默认情况下为 10 秒）
   - 在第一个请求生成错误时，源返回的 `Cache-Control max-age` 标头或 `Cache-Control s-maxage` 标头指定的时间量
2. 对于缓存持续时间（在第 1 步中确定），CloudFront 会以缓存的 4xx 或 5xx 状态代码来响应针对同一对象的后续查看方请求。

3. 经过了缓存时间（在第 1 步中确定）之后，CloudFront 向源转发另一个请求以再次尝试获取请求的对象。CloudFront 将继续按照由最短错误缓存 TTL 指定的间隔重试。

请求的对象在边缘缓存中

在满足以下所有条件时，CloudFront 将继续提供当前在边缘缓存中的对象：

- 查看器请求对象。
- 对象在边缘缓存中但已过期。
- 源返回 HTTP 5xx 状态代码，而不是返回 304 状态代码（未修改）或更新版本的对象。

CloudFront 将执行以下操作：

1. 如果您的源返回 5xx 错误代码，则 CloudFront 将提供对象，即使它已过期。在错误缓存最小 TTL 的持续时间（默认情况下为 10 秒）内，CloudFront 会通过从边缘缓存提供对象来继续响应查看器请求。

   如果您的源返回一个 4xx 状态代码，则 CloudFront 会将该状态代码而不是请求的对象返回给查看器。
2. 在最短错误缓存 TTL 过后，CloudFront 会通过再向您的源转发一个请求来再次尝试获取请求的对象。请注意，如果未频繁请求该对象，当您的源服务器仍返回 5xx 响应时，CloudFront 可能会将该对象从边缘缓存中逐出。有关对象在 CloudFront 边缘缓存中的保留时长的信息，请参阅管理内容保留在缓存中的时间长度（过期）。

CloudFront 缓存的 HTTP 4xx 和 5xx 状态代码

CloudFront 根据返回的具体状态代码以及源是否在响应中返回了特定标头，缓存您的源返回的 4xx 和 5xx HTTP 状态代码。

CloudFront 始终缓存的 HTTP 4xx 和 5xx 状态代码

CloudFront 始终缓存您的源返回的以下 HTTP 4xx 和 5xx 状态代码。如果您已针对某一 HTTP 状态代码配置自定义错误页面，则 CloudFront 会缓存该自定义错误页面。

404	Not Found

414	Request-URI Too Large
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	网关超时

CloudFront 根据 **Cache-Control** 标头缓存的 HTTP 4xx 的状态代码

如果源返回 `Cache-Control max-age` 或 `Cache-Control s-maxage` 标头，则 CloudFront 仅缓存源返回的以下 HTTP 4xx 状态代码。如果您已为这些 HTTP 状态代码之一配置了自定义错误页面，并且您的源返回了缓存控制标头之一，CloudFront 缓存自定义错误页面。

400	Bad Request
403	禁止
405	Method Not Allowed
412[1]	前置条件失败
415[1]	媒体类型不受支持

[1]CloudFront 不支持为这些 HTTP 状态代码创建自定义错误页面。

生成自定义错误响应

如果您通过 CloudFront 提供的对象出于某种原因不可用，您的 Web 服务器通常会返回相应的 HTTP 状态代码到 CloudFront 来进行指示。例如，如果查看器请求了无效 URL，您的 Web 服务器将向 CloudFront 返回 HTTP 404（未找到）状态代码，然后 CloudFront 会将此状态代码返回至查看器。您可以不使用此默认错误响应，而是创建一个由 CloudFront 返回给查看器的自定义错误响应。

如果您配置 CloudFront 来为 HTTP 状态代码返回自定义错误页面，但自定义错误页面不可用，则 CloudFront 将向查看器返回 CloudFront 从包含自定义错误页面的源接收的状态代码。例如，假设您的自定义源返回 500 状态代码且您已将 CloudFront 配置为从 Amazon S3 存储桶获取 500 状态代码的自定义错误页面。但是，某人意外删除了您 Amazon S3 存储桶中的自定义错误页面。CloudFront 将 HTTP 404 状态代码（未找到）返回给请求对象的查看器。

当 CloudFront 将自定义错误页面返回至查看器时，您支付的是自定义错误页面的标准 CloudFront 费用，而不是所请求对象的费用。有关 CloudFront 费用的更多信息，请参阅 Amazon CloudFront 定价。

主题

配置错误响应行为

您可以通过多个选项，管理 CloudFront 在出错时如何进行响应。要配置自定义错误响应，您可以使用 CloudFront 控制台、CloudFront API 或 AWS CloudFormation。无论您选择哪种方式来更新配置，请考虑以下提示和建议：

- 在 CloudFront 可访问的位置保存自定义错误页面。建议您将这些页面存储在 Amazon S3 存储桶中，并且不要将它们与您的网站或应用程序的其余内容存储于同一位置。如果您将自定义错误页面与您的网站或应用程序存储在同一个源上，且源开始返回 5xx 错误，则 CloudFront 无法获取自定义错误页面，因为源服务器不可用。有关更多信息，请参阅将对象和自定义错误页面存储在不同的位置。
- 确保 CloudFront 有权获取自定义错误页面。如果自定义错误页面存储在 Amazon S3 中，则这些页面必须可公开访问，或者您必须配置 CloudFront 源访问控制（OAC）。如果自定义错误页面存储在自定义源中，则这些页面必须可公开访问。

- （可选）如果您需要，还可以配置您的源，以添加 `Cache-Control` 或 `Expires` 标头以及自定义错误页面。您还可以使用错误缓存最短 TTL 设置来控制 CloudFront 缓存自定义错误页面的时长。有关更多信息，请参阅 控制 CloudFront 缓存错误的时间长度。

配置自定义错误响应

要在 CloudFront 控制台中配置自定义错误响应，您必须具有 CloudFront 分配。在控制台中，自定义错误响应的配置设置仅适用于现有分配。要了解如何创建分配，请参阅 开始使用基本 CloudFront 分配。

Console

配置自定义错误响应（控制台）

1. 登录到AWS Management Console并通过以下网址在 CloudFront 控制台 (https://console.aws.amazon.com/cloudfront/v4/home#distributions) 中打开分配页面。
2. 在分配列表里，选择要更新的分配。
3. 选择错误页面选项卡，然后选择创建自定义错误响应。
4. 输入适用的值。有关更多信息，请参阅 自定义错误页面和错误缓存。
5. 输入所需的值后，选择创建。

CloudFront API or AWS CloudFormation

要通过 CloudFront API 或 AWS CloudFormation 配置自定义错误响应，您可以使用分配中的 `CustomErrorResponse` 类型。有关更多信息，请参阅下列内容：

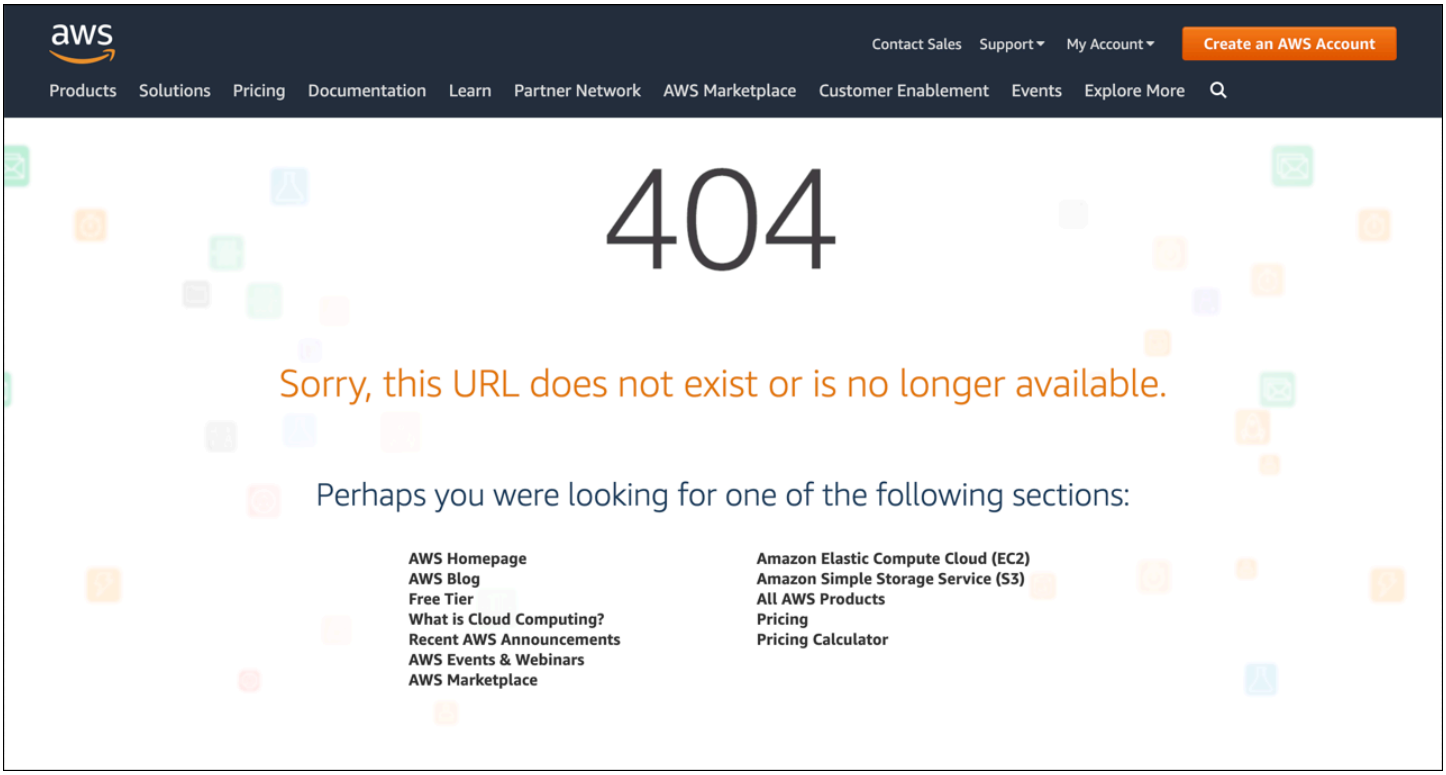
- 《AWS CloudFormation 用户指南》 中的 AWS::CloudFront::Distribution CustomErrorResponse
- Amazon CloudFront API 参考中的 CustomErrorResponse

为特定 HTTP 状态代码创建自定义错误页面

如果您希望显示自定义错误消息而不是默认消息 – 例如，使用与网站其他部分相同的格式设置的页面 – 则可以让 CloudFront 向查看器返回包含自定义错误消息的对象（例如，HTML 文件）。

要指定您希望返回的文件以及应为此文件返回的错误，请更新您的 CloudFront 分配以指定这些值。有关更多信息，请参阅配置错误响应行为。

例如，以下是一个自定义错误页面：



您可以为每个支持的 HTTP 状态代码指定一个不同的对象，也可以对所有支持的状态代码使用同一个对象。您可以选择为一些状态代码指定自定义错误页面，而不为另外一些状态代码指定自定义错误页面。

您通过 CloudFront 提供的对象出于各种原因可能不可用。可将这些原因归为以下两大类：

- 客户端错误表示请求出现问题。例如，具有指定名称的对象不可用，或用户不具有获取您 Amazon S3 存储桶中的对象所需的权限。当出现客户端错误时，源会向 CloudFront 返回一个 4xx 区间内的 HTTP 状态代码。
- 服务器错误表示源服务器出现问题。例如，HTTP 服务器繁忙或不可用。当出现服务器错误时，您的源服务器将向 CloudFront 返回一个 5xx 区间内的 HTTP 状态代码，或者在某一时间段内 CloudFront 未从您的源服务器获得响应，此时显示 504 状态代码（网关超时）。

CloudFront 可以为其返回自定义错误页面的 HTTP 状态代码包含以下各项：

- 400、403、404、405、414、416

> **注意**
>
> - 如果 CloudFront 检测到请求可能不安全，CloudFront 将返回 400（错误请求）错误，而不是自定义错误页面。
> - 您可以为 HTTP 状态代码 416（无法满足请求的范围）创建自定义错误页面，并可以更改源向 CloudFront 返回状态代码 416 时 CloudFront 向查看器返回的 HTTP 状态代码。（有关更多信息，请参阅 更改 CloudFront 返回的响应代码。）但是，CloudFront 不缓存状态代码 416 响应，因此，即使您为状态代码 416 指定错误缓存最小 TTL 的值，CloudFront 也不会使用它。

- 500、501、502、503、504

> **Note**
>
> 在某些情况下，即使您对 CloudFront 进行配置，使其为 HTTP 503 状态代码返回自定义错误页面，CloudFront 也不会执行此操作。如果 CloudFront 错误代码为 `Capacity Exceeded` 或 `Limit Exceeded`，CloudFront 会将 503 状态代码（而非自定义错误页面）返回至查看器。

有关 CloudFront 如何处理来自您的源的错误响应的详细说明，请参阅CloudFront 如何处理来自源的 HTTP 4xx 和 5xx 状态代码。

将对象和自定义错误页面存储在不同的位置

如果您希望将您的对象和自定义错误页面存储在不同的位置，您的分配必须包含满足以下条件时的缓存行为：

- Path Pattern 的值与您的自定义错误消息的路径匹配。例如，假设您在 Amazon S3 存储桶中名为 `/4xx-errors` 的目录下为 4xx 错误保存了自定义错误页面。您的分配必须包含缓存行为，其路径模式将对自定义错误页面的请求路由至该位置，例如 `/4xx-errors/*`。
- 源值指定包含您的自定义错误页面的源的源 ID 值。

有关更多信息，请参阅 缓存行为设置。

更改 CloudFront 返回的响应代码

您可以配置 CloudFront 向查看器返回的 HTTP 状态代码，使其与 CloudFront 从源接收的状态代码不相同。例如，如果源向 CloudFront 返回 500 状态代码，您可能希望 CloudFront 向查看器返回自定义错误页面和 200 状态代码（正常）。出于多种原因，您可能希望 CloudFront 向查看器返回不同于源返回到 CloudFront 的状态代码：

- 一些互联网设备（例如，一些防火墙和企业代理）会拦截 HTTP 4xx 和 5xx 状态代码，防止响应返回到查看器。在这种情况下，如果您替换 200，就不会截获响应。
- 如果您不希望区分不同的客户端错误或服务器错误，可以将 CloudFront 为所有 4xx 或 5xx 状态代码返回的值指定为 400 或 500。
- 您可能希望返回 200 状态代码（正常）和静态网站，确保客户不知道网站宕机。

如果您启用 CloudFront 标准日志并且将 CloudFront 配置为更改响应中的 HTTP 状态代码，则标准日志中 sc-status 列的值将包含您指定的状态代码。但是，x-edge-result-type 列的值不受影响。该值包含源响应的结果类型。例如，假设您将 CloudFront 配置为在源将 200（未找到）返回到 CloudFront 时将 404 的状态代码返回到查看器。当源使用 404 状态代码响应请求时，日志中 sc-status 列的值将为 200，但 x-edge-result-type 列的值将为 Error。

您可以将 CloudFront 配置为随自定义错误页面返回以下任意 HTTP 状态代码：

- 200
- 400、403、404、405、414、416
- 500、501、502、503、504

控制 CloudFront 缓存错误的时间长度

CloudFront 缓存错误响应的默认持续时间为 10 秒。CloudFront 然后将对象的下一个请求提交给您的源，以查看导致错误的问题是否已解决，并且请求的对象是否可用。

您可以为 CloudFront 缓存的各 4xx 和 5xx 状态代码指定错误缓存时长（错误缓存最小 TTL）。（有关更多信息，请参阅 CloudFront 缓存的 HTTP 4xx 和 5xx 状态代码。）指定持续时间时，注意以下几点：

- 如果您指定一个很短的错误缓存时长，则与指定较长的时长相比，CloudFront 将向您的源转发更多请求。对于 5xx 错误，这可能会加重原先导致源返回错误的问题。

- 当源针对某个对象返回错误时，CloudFront 会通过错误响应或自定义错误页面响应对象请求，直到超过错误缓存时间长度。如果您指定一个很长的错误缓存持续时间，CloudFront 可能会在对象再次转为可用后的很长一段时间内继续使用错误响应或您的自定义错误页面来响应请求。

> **Note**
>
> 您可以为 HTTP 状态代码 416（无法满足请求的范围）创建自定义错误页面，并可以更改源向 CloudFront 返回状态代码 416 时 CloudFront 向查看器返回的 HTTP 状态代码。（有关更多信息，请参阅 更改 CloudFront 返回的响应代码。）但是，CloudFront 不缓存状态代码 416 响应，因此，即使您为状态代码 416 指定错误缓存最小 TTL 的值，CloudFront 也不会使用它。

如果您要控制 CloudFront 为各个对象缓存错误的时间长度，可以配置源服务器以为该对象的错误响应添加相应标头。

如果源添加 `Cache-Control: max-age` 或 `Cache-Control: s-maxage` 指令或 `Expires` 标头，则 CloudFront 缓存错误响应的时间为标头中的值或错误缓存最小 TTL 值（以较大的值为准）。

> **Note**
>
> `Cache-Control: max-age` 和 `Cache-Control: s-maxage` 值不能大于为提取错误页面的缓存行为设置的最大 TTL 值。

如果源添加其他 `Cache-Control` 指令或不添加标头，则 CloudFront 缓存错误响应的时间等于错误缓存最小 TTL 的值。

如果某一对象的 4xx 或 5xx 状态代码的过期时间超过您希望等待的时间且对象再次可访问，您可以使用所请求对象的 URL 使缓存错误代码失效。如果源返回针对多个对象的错误响应，您需要分别使各个对象失效。有关使对象失效的更多信息，请参阅 使文件失效以删除内容。

添加、删除或替换 CloudFront 分配的内容

此部分介绍了如何确保 CloudFront 能够访问要提供给查看器的内容，如何指定网站或应用程序中的对象以及如何删除或替换内容。

主题

添加和访问 CloudFront 分配的内容

如果您希望 CloudFront 分配内容（对象），则可以将文件添加到为分配指定的源之一，然后公开指向这些文件的 CloudFront 链接。CloudFront 边缘站点不从源提取新文件，直至边缘站点收到了针对这些文件的查看器请求。有关更多信息，请参阅 CloudFront 如何交付内容。

在您添加希望 CloudFront 分配的文件时，请确保将其添加到您在分配中指定的 Amazon S3 存储桶之一，对于自定义源，将其添加到指定域中的目录。此外，确认适用缓存行为中的路径模式将请求发送到正确的源。

例如，假设缓存行为的路径模式是 `*.html`。如果您尚未配置任何其他缓存行为以将请求转发到源，则 CloudFront 仅转发 `*.html` 文件。在这种情况下，例如，CloudFront 绝不会分发您上传到源的 .jpg 文件，因为您尚未创建包括 .jpg 文件的缓存行为。

CloudFront 服务器不确定他们提供的对象的 MIME 类型。当您将一个文件上传到源时，建议您为该文件设置 `Content-Type` 标头字段。

使用文件版本控制，通过 CloudFront 分配来更新或删除内容

要更新设置为通过 CloudFront 分配给您的现有内容，建议您在文件名或文件夹名称中使用版本标识符。这有助于您更好地管理 CloudFront 提供的内容。

使用版本化文件名更新现有文件

当您更新 CloudFront 分配中的现有文件时，建议您在文件名或目录名称中包括某种版本的标识符，以便更好地控制自己的内容。此标识符可能是日期时间戳、序列号、或区别同一对象的两种版本的其他方法。

例如，取代命名图像文件 image.jpg，您可称之为 image_1.jpg。当您想开始提供新版本的文件时，您需要将新文件命名为 image_2.jpg，并且更新 Web 应用程序或网站中的链接以指向 image_2.jpg。此外，您可将所有的图形放在 image_v1 目录中，且当您想开始提供一个或多个图像的新版本时，您会创建新的 image_v2 目录，并且您会更新指向该目录的链接。凭借版本控制，您不必在 CloudFront 开始提供对象新版本之前等待期过期，并且您不必为对象的失效支付费用。

即使您对文件进行版本控制，我们仍建议您设置到期日期。有关更多信息，请参阅 管理内容保留在缓存中的时间长度（过期）。

> **Note**
>
> 指定版本控制的文件名或目录名与 Amazon S3 对象版本控制无关。

删除内容，这样 CloudFront 不会再分发该内容

您可以从源中删除不再希望包含在 CloudFront 分配中的文件。但是，CloudFront 将继续为查看器显示边缘缓存中的内容，直至文件过期。

如果您要立即删除文件，您必须执行下列操作之一：

- 使用文件版本控制。当您使用版本控制时，不同版本的文件具有可在 CloudFront 分配中使用的不同名称，以便更改返回给查看器的文件。有关更多信息，请参阅 使用版本化文件名更新现有文件。
- 使文件失效。有关更多信息，请参阅 使文件失效以删除内容。

在 CloudFront 中自定义文件的 URL 格式

在为源设置您希望 CloudFront 提供给查看器的对象（内容）后，您必须使用正确的 URL 在网站或应用程序代码中引用这些对象，以便 CloudFront 可以提供该对象。

您在网页或 Web 应用程序中的对象的 URL 中使用的域名可以是以下任一种：

- CloudFront 在您创建分配时自动分配的域名，例如 `d111111abcdef8.cloudfront.net`

- 您自己的域名，例如 example.com

例如，您可以使用以下 URL 之一来返回文件 image.jpg：

https://d111111abcdef8.cloudfront.net/images/image.jpg

https://example.com/images/image.jpg

可使用相同的 URL 格式，无论您将内容存储在 Amazon S3 存储桶中还是自定义源中，如您自己的某台 Web 服务器。

> **Note**
>
> URL 格式部分取决于您在分配中为源路径指定的值。此值为 CloudFront 提供了您的对象的顶级目录路径。有关在您创建分配时设置源路径的更多信息，请参阅源路径。

有关 URL 格式的更多信息，请参阅下面几节。

使用您自己的域名（example.com）

您可以不使用在创建分配时 CloudFront 为您分配的默认域名，而是添加更易于使用的备用域名，例如 example.com。通过使用 CloudFront 设置您自己的域名，您可以对分配中的对象使用如下 URL：

https://example.com/images/image.jpg

如果您计划在查看器和 CloudFront 之间使用 HTTPS，请参阅使用备用域名和 HTTPS。

在 URL 中使用尾随斜杠（/）

在您为 CloudFront 分配中的目录指定 URL 时，选择始终使用尾随斜杠或从不使用尾随斜杠。例如，为您的所有 URL 仅选择以下格式之一：

https://d111111abcdef8.cloudfront.net/images/

https://d111111abcdef8.cloudfront.net/images

这为什么非常重要？

这两种格式都可用于链接到 CloudFront 对象，但保持一致有助于防止您稍后要使目录失效时出现问题。CloudFront 完全按照 URL 定义的样子进行存储，包括尾随斜杠。因此，如果格式不一致，您将需要分别使带和不带斜杠的目录 URL 失效，以确保 CloudFront 删除该目录。

必须使两种 URL 格式无效很不方便，并且会引发额外成本。这是因为，如果您必须执行两次失效操作来涵盖两种类型的 URL，则您可能会超过当月的最大免费失效次数。如果出现这种情况，您将必须为所有失效付费，即使 CloudFront 中仅存在每个目录 URL 的一种格式也是如此。

为限制内容创建签名 URL

如果您要限制对您的内容的访问，可以创建签名 URL。例如，如果您希望将内容仅分发到经过身份验证的用户，则可以创建仅在指定时间段内有效或只能从指定 IP 地址使用的 URL。有关更多信息，请参阅 使用签名 URL 和签名 Cookie 提供私有内容。

指定默认根对象

当用户请求分配的根 URL 而不是请求分配中的对象时，您可配置 CloudFront 以返回特定的对象（默认根对象）。通过指定一个默认根对象，您可以避免公开分配的内容。

主题

如何指定默认根对象

要避免公开分配的内容或返回错误，请通过以下步骤来为您的分配指定默认根对象。

要为分配指定默认根对象

1. 将默认根对象上传到分配指向的源。

   文件可为 CloudFront 支持的任何类型。对于文件名的约束列表，请参阅 `DefaultRootObject`DistributionConfig 中的元素说明。

   > **Note**
   >
   > 如果默认根对象的文件名太长或包含无效字符，CloudFront 则返回错误 `HTTP 400 Bad Request - InvalidDefaultRootObject`。此外，CloudFront 将缓存代码 10 秒钟（默认情况下），并将结果写入访问日志。

2. 确认对象的权限至少授予 CloudFront `read` 访问权。

有关 Amazon S3 权限的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的 Amazon S3 中的标识和访问权限管理。

3. 更新分配，以使用 CloudFront 控制台或 CloudFront API 关连默认根对象。

   要使用 CloudFront 控制台指定默认根对象：

   a. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
   b. 在顶部窗格中的分配列表里，选择要更新的分配。
   c. 在设置窗格中的常规选项卡上，选择编辑。
   d. 在编辑设置对话框的默认根对象字段中，输入默认根对象的文件名。

      仅输入对象名称，例如 `index.html`。不要在对象名称前添加 `/`。
   e. 选择保存更改。

   要使用 CloudFront API 更新您的分配，请在您的分配中指定 `DefaultRootObject` 元素的值。有关使用 CloudFront API 指定默认根对象的信息，请参阅 Amazon CloudFront API 参考中的 UpdateDistribution。
4. 确认您已经通过请求根 URL 启用了默认根对象。如果您的浏览器不显示默认根对象，请执行以下步骤：

   a. 通过在 CloudFront 控制台查看分配的状态，确认您的分配已经完全部署。
   b. 重复第 2 步和第 3 步，确认您已授予正确的权限，并且您已正确地更新分配的配置来指定默认根对象。

原定设置根对象的工作原理

假定以下请求指向对象 `image.jpg`：

```
https://d111111abcdef8.cloudfront.net/image.jpg
```

相反，以下请求指向相同分配的根 URL 而不是特定的对象，如第一个示例中所述：

```
https://d111111abcdef8.cloudfront.net/
```

当您定义默认根对象时，调用分配的根的最终用户请求返回默认根对象。例如，如果您指定文件 `index.html` 作为您的默认根对象，请求：

`https://d111111abcdef8.cloudfront.net/`

返回值:

`https://d111111abcdef8.cloudfront.net/index.html`

> **Note**
>
> CloudFront 不确定带有多个尾部斜杠的 URL（`https://d111111abcdef8.cloudfront.net///`）是否等同于 `https://d111111abcdef8.cloudfront.net/`。您的原始服务器进行该比较。

如果您定义原定设置根对象，最终用户对于分配的子目录的请求不返回该原定设置根对象。例如，假设 `index.html` 是您的默认根对象且 CloudFront 接收最终用户对 CloudFront 分配下的 `install` 目录的请求：

`https://d111111abcdef8.cloudfront.net/install/`

CloudFront 不会返回默认根对象，即使 `index.html` 的副本出现在 `install` 目录中。

如果您将自己的分配配置为允许 CloudFront 支持的所有 HTTP 方法，则默认根对象适用于所有方法。例如，如果您的默认根对象为 index.php 并且您编写应用程序以将 POST 请求提交到您的根域 (https://example.com)，则 CloudFront 会将请求发送到 https://example.com/index.php。

CloudFront 默认根对象的行为与 Amazon S3 索引文档的行为不同。当您配置 Amazon S3 存储桶作为网站并指定索引文档时，Amazon S3 将返回索引文档，即使用户请求存储桶中的子目录。（索引文档副本必须出现在每个子目录中。）有关配置 Amazon S3 存储桶作为网站以及索引文档的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的在 Amazon S3 上托管网站章节。

> **Important**
>
> 请记住，默认根对象仅适用于 CloudFront 分配。您仍需要管理源的安全。例如，如果您使用 Amazon S3 源，仍需要适当地设置您的 Amazon S3 存储桶 ACL，以确保您想要的存储桶访问级别。

未定义根对象的情况下 CloudFront 的工作方式

如果您不定义默认根对象，对分配的根请求则传递到源服务器。如果您使用 Amazon S3 源，则可能返回以下任何内容：

- Amazon S3 存储桶的内容列表 – 在以下任何条件下，使用 CloudFront 访问分配的任何人均可看见源的内容：
  - 您的存储桶未正确配置。
  - 与分配有关的存储桶和存储桶中对象的 Amazon S3 权限授予每个人访问权。
  - 最终用户使用源的根 URL 访问您的源。
- 源的私有内容列表 – 如果您配置您的源作为私有分配（仅您和 CloudFront 有访问权），具有凭证通过 CloudFront 访问您的分配的任何人可看见与您的分配有关的 Amazon S3 存储桶的内容。在这种情况下，用户不能通过源的根 URL 访问您的内容。更多有关分配私有内容的信息，请参阅 the section called “使用签名 URL 和签名 Cookie 限制内容”。
- `Error 403 Forbidden` – 如果与分配有关的 Amazon S3 存储桶上的权限或该存储桶中对象上的权限拒绝 CloudFront 和每个人访问，CloudFront 则返回该错误。

使文件失效以删除内容

如果您需要在文件过期前从 CloudFront 边缘缓存中删除文件，可以执行下列操作之一：

- 通过边缘缓存使文件失效。查看器下次请求文件时，CloudFront 将返回源以获取文件的最新版本。
- 使用文件版本控制以提供具有不同名称的文件的不同版本。有关更多信息，请参阅 使用版本化文件名更新现有文件。

主题

在使文件失效和使用版本控制的文件名之间进行选择

要控制从分配提供的文件版本，您可使文件失效或使用版本控制文件名为其命名。如果您希望频繁地更新文件，出于以下原因，建议您主要使用文件版本控制：

- 即使用户在本地或企业缓存代理中缓存了文件的版本，版本控制也使您能够控制为请求返回哪个文件。如果您使文件失效，用户看到的可能继续是旧版本直至它从这些缓存中过期。
- CloudFront 访问日志包括文件的名称，因此，版本控制使分析文件变更的结果变得更加轻松。
- 版本控制提供一种将不同版本的文件提供给不同用户的方式。
- 版本控制简化了文件修订之间的向前和向后滚动。
- 版本控制更便宜。您仍需为 CloudFront 支付费用以将文件的新版本传输到边缘站点，但不必为使文件失效支付费用。

有关文件版本控制的更多信息，请参阅使用版本化文件名更新现有文件。

确定使哪个文件失效

如果您要使多个文件失效，例如某个目录中的所有文件或者名称以相同字符开头的所有文件，您可以在失效路径的结尾包含 * 通配符。有关使用 * 通配符的更多信息，请参阅 Invalidation paths。

要使文件失效，您可以指定单独文件的路径或以 * 通配符结尾的路径，后者可能会应用到一个或多个文件，如以下示例中所示：

- `/images/image1.jpg`
- `/images/image*`
- `/images/*`

如果您想使选定文件失效但用户不必访问源上的每个文件，可确定查看器从 CloudFront 请求了哪些文件并只使这些文件失效。要决定查看器已经请求的文件，请启用 CloudFront 访问日志记录。有关访问日志的更多信息，请参阅 配置和使用标准日志（访问日志）。

使文件失效时的需知事项

指定要使其失效的文件时，请参考以下信息：

区分大小写

失效路径区分大小写。例如，`/images/image.jpg` 和 `/images/Image.jpg` 指定两个不同的文件。

使用 Lambda 函数更改 URI

如果您的 CloudFront 分配在发生查看器请求事件时触发 Lambda 函数，并且该函数更改请求的文件的 URI，建议您使这两个 URI 失效，以便从 CloudFront 边缘缓存中删除该文件：

- 查看器请求中的 URI
- 函数进行更改后的 URI

Example 示例

假设您的 Lambda 函数将某个文件的 URI 从：

```
https://d111111abcdef8.cloudfront.net/index.html
```

更改为包含语言目录的 URI：

```
https://d111111abcdef8.cloudfront.net/en/index.html
```

要使该文件失效，您必须指定以下路径：

- `/index.html`
- `/en/index.html`

有关更多信息，请参阅 Invalidation paths。

默认根对象

要使默认根对象（文件）失效，请以您为任何其他文件指定路径相同的方式指定路径。有关更多信息，请参阅 原定设置根对象的工作原理。

转发 Cookie

如果您将 CloudFront 配置为将 Cookie 转发到您的源，CloudFront 边缘缓存可能会包含文件的多个版本。当您使文件失效之后，CloudFront 会使文件的所有缓存版本失效，不论其关联的 Cookie 如何。您无法根据关联的 Cookie 选择性地使一些版本失效，而使另一些版本不失效。有关更多信息，请参阅 根据 Cookie 缓存内容。

转发标头

如果您将 CloudFront 配置为将标头列表转发到源并基于标头的值进行缓存，CloudFront 边缘缓存可能会包含文件的多个版本。当您使文件失效之后，CloudFront 会使文件的所有缓存版本失效，而不论标头值如何。您无法根据标头值选择性地使一些版本失效，而使另一些版本不失效。（如果您将 CloudFront 配置为将所有标头转发到源，则 CloudFront 不会缓存您的文件。）有关更多信息，请参阅 根据请求标头缓存内容。

转发查询字符串

如果您将 CloudFront 配置为将查询字符串转发到您的源，在使文件失效时，必须包括查询字符串，如下例中所示：

- `/images/image.jpg?parameter1=a`
- `/images/image.jpg?parameter1=b`

如果客户端请求包括针对同一个文件的五个不同查询字符串，您可以使文件失效五次（每个查询字符串一次），或者您可以在失效路径中使用 * 通配符，如下例中所示：

```
/images/image.jpg*
```

有关在失效路径中使用通配符的更多信息，请参阅 Invalidation paths。

有关查询字符串的更多信息，请参阅 根据查询字符串参数缓存内容。

要确定哪个查询字符串正在使用中，您可启用 CloudFront 日志记录。有关更多信息，请参阅 配置和使用标准日志（访问日志）。

允许的最大值

有关允许的最大失效次数的信息，请参阅并发失效请求最大值。

Microsoft Smooth Streaming 文件

为对应的缓存行为启用 Smooth Streaming 时，您不能使 Microsoft Smooth Streaming 格式的媒体文件失效。

路径中的非 ASCII 或不安全字符

如果路径包含非 ASCII 字符或 RFC 1738 中定义的不安全字符，请对这些字符进行 URL 编码。切勿对路径中的任何其他字符进行 URL 编码，否则，CloudFront 将不会使已更新文件的旧版本失效。

失效路径

路径是相对于分配的。例如，要使 `https://d111111abcdef8.cloudfront.net/images/image2.jpg` 处的对象失效，您可指定 `/images/image2.jpg`。

> **Note**
>
> 在 CloudFront 控制台中，您可以省略路径中的前导斜杠，如下所示：`images/image2.jpg`。在直接使用 CloudFront API 时，失效路径必须以前导斜杠开头。

您还可以使用 * 通配符使多个文件同时失效。* 将替换 0 个或多个字符，必须是失效路径中最后的字符。

如果您使用 AWS Command Line Interface（AWS CLI）使文件失效，并且指定一个包含 * 通配符的路径，则必须使用引号（"）将路径引起来，例如 "/*"。

Example 示例：使路径失效

- 使目录中的所有文件失效：

  */directory-path/**

- 使目录、其所有子目录以及该目录和子目录中的所有文件失效：

  */directory-path**

- 使具有相同名称但文件扩展名不同的所有文件失效 (例如 logo.jpg、logo.png 和 logo.gif)：

  */directory-path/file-name.**

- 使目录中文件名以相同字符开头的所有文件失效 (例如 HLS 格式的视频的所有文件) 而不论扩展名如何：

  */directory-path/initial-characters-in-file-name**

- 当您将 CloudFront 配置为基于查询字符串参数进行缓存并且希望使某个文件的所有版本失效时：

  */directory-path/file-name.file-name-extension**

- 使分配中的所有文件失效：

  /*

路径的长度上限是 4000 个字符。您不能在路径内使用通配符，只能在路径终点处添加通配符。

有关使用 Lambda 函数更改 URI 时使文件失效的信息，请参阅Changing the URI Using a Lambda Function。

如果失效路径为目录且您对指定目录的方法尚未标准化 (带或不带尾部斜杠 /)，建议您使带和不带尾部斜杠的目录均失效，例如，/images 和 /images/。

已签名的 URL

如果您正在使用签名 URL，可以通过在问号 (?) 前仅包括部分 URL 来使文件失效。

使文件失效

您可以使用 CloudFront 控制台来创建并运行失效、显示您先前提交的失效列表以及有关单个失效的详细信息。您还可复制现有的失效、编辑文件路径列表并运行已编辑的失效。您无法从列表中删除失效。

目录

使文件失效

要使用 CloudFront 控制台使文件失效，请执行以下操作。

Console

使文件失效（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择您想为其使文件失效的分配。
3. 选择失效选项卡。
4. 选择创建失效。
5. 对于您希望使其失效的文件，请在每行输入一个失效路径。有关指定失效路径的信息，请参阅使文件失效时的需知事项。

   > **Important**
   >
   > 请谨慎指定文件路径。在启动之后，您无法取消失效请求。

6. 选择创建失效。

CloudFront API

要了解如何使对象失效以及显示失效相关信息，请参阅《Amazon CloudFront API 参考》中的以下主题：

- CreateInvalidation
- ListInvalidations

- GetInvalidation

> **Note**
>
> 如果您使用 AWS Command Line Interface（AWS CLI）使文件失效，并且指定了一个包含 * 通配符的路径，则必须使用引号（"）将路径引起来，如以下所示：
>
> ```
> aws cloudfront create-invalidation --distribution-id distribution_ID --paths
>  "/*"
> ```

复制、编辑和重新运行现有失效

您可复制您先前创建的失效、更新失效路径列表以及运行已更新的失效。您不可复制现有失效、更新失效路径，然后在不运行已更新失效的情况下将其保存。

> **Important**
>
> 如果您复制仍在进行中的失效、更新失效路径列表，然后运行已更新的失效，则 CloudFront 不会停止或删除您复制的失效。如果任何失效路径出现在原本和副本中，CloudFront 将尝试使文件失效两次，且两次失效均计入到您每月免费失效的最大数目中。如果您已经达到了免费失效的最大数目，将会向您收取每个文件两次失效的费用。有关更多信息，请参阅 并发失效请求最大值。

复制、编辑及重新运行现有失效

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择包含要复制的失效的分配。
3. 选择失效选项卡。
4. 选择您想复制的失效。

   如果您不确定要复制哪个失效，可以选择某个失效，然后选择查看详细信息以显示该失效的详细信息。
5. 选择复制到新项目。

6. 更新失效路径列表 (如适用)。
7. 选择创建失效。

取消失效

当您将失效请求提交到 CloudFront 时，CloudFront 会在几秒钟内将该请求转发到所有边缘站点，每个边缘站点将立即开始处理失效。因此，一旦提交失效便无法取消它。

列出失效

通过使用 CloudFront 控制台，您可显示您已为分配创建和运行的最后 100 个失效的列表。如果您想获得超过 100 个失效的列表，请执行 `ListInvalidations` API 操作。有关更多信息，请参阅《Amazon CloudFront API 参考》中的 ListInvalidations。

列出失效

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择您要为其显示失效列表的分配。
3. 选择失效选项卡。

> **Note**
>
> 您无法从列表中删除失效。

显示有关失效的信息

您可显示失效的详细信息，包括分配 ID、失效 ID、失效状态、创建失效的日期和时间以及完整的失效路径列表。

显示有关失效的信息

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择包含您要为其显示详细信息的失效的分配。
3. 选择失效选项卡。

4. 选择适用的失效 ID 或选择失效 ID，然后选择查看详细信息。

并发失效请求最大值

如果您想逐个让文件失效，则针对每个分配一次可以同时处理 3000 个文件的无效请求。一个无效请求可以支持最多 3000 个文件，或针对一个文件的 3000 个请求，或者合计不超过 3000 个文件的任何组合方式。例如，您可以提交 30 个失效请求，每个请求使 100 个文件失效。只要全部 30 个失效请求仍在进行中，您无法继续提交任何失效请求。如果超出最大值，CloudFront 将返回一条错误消息。

如果您使用 * 通配符，则一次最多可以提出 15 个无效路径请求。您一次可以为每个分配的最多 3000 个单独文件提出失效请求，而允许的通配符无效请求的最大值与个别使文件失效的最大值无关。

支付文件失效费用

您每月提交的前 1000 个失效路径免费；每月超过 1000 个之后，您须为每个失效路径支付费用。一个失效路径可以针对单个文件（如 `/images/logo.jpg`），也可针对多个文件（如 `/images/*`）。一个包括 * 通配符的路径计数为一个路径，即使该路径导致 CloudFront 使数千个文件失效。

每月 1000 个免费失效路径的最大值适用于您用一个 AWS 账户创建的所有分配的失效路径总数。例如，如果您使用 AWS 账户 `john@example.com` 创建了三个分配，并在指定月份里为每个分配提交了 600 个失效路径（总共 1800 个失效路径），那么 AWS 在该月将向您收取 800 个失效路径的费用。

不论您要使之失效的文件的数量是多少（单个文件 (`/images/logo.jpg`) 或是与分配关联的所有文件 (`/*`)），提交失效路径的费用均相同。由于在失效请求中，您需要按路径付费，因此即使您将多个路径捆绑到一个请求中，出于计费目的，每个路径仍会单独计数。

有关失效定价的更多信息，请参阅 Amazon CloudFront 定价。有关失效路径的更多信息，请参阅 Invalidation paths。

提供压缩文件

您可以使用 CloudFront 自动压缩某些类型的对象（文件），并在查看器（Web 浏览器或其他客户端）支持压缩对象时（Web 浏览器或其他客户端）提供这些压缩对象。查看器通过 `Accept-Encoding` HTTP 标头指示它们支持压缩对象。

CloudFront 可以使用 Gzip 和 Brotli 压缩格式压缩对象。当查看器支持这两种格式，并且两者都存在于所访问的缓存服务器中时，CloudFront 会首选 Brotli。如果缓存服务器中只有一种压缩格式，CloudFront 会将其返回。

> **Note**
>
> Chrome 和 Firefox Web 浏览器仅在使用 HTTPS 发送请求时支持 Brotli 压缩。这些浏览器不支持带 HTTP 请求的 Brotli。

在压缩请求的对象时，由于对象更小（在某些情况下，大小不到原件的四分之一），因此下载速度可能更快。特别是对于 JavaScript 和 CSS 文件，更快的下载会导致向用户更快地提供网页。此外，由于 CloudFront 数据传输的费用基于提供的数据总量，因此提供压缩对象的成本将低于提供未压缩的对象。

某些自定义源也可以压缩对象。您的源也许能够压缩 CloudFront 无法压缩的对象（请参阅 CloudFront 压缩的文件类型）。如果您的源将压缩对象返回到 CloudFront，则 CloudFront 将根据是否存在 `Content-Encoding` 标头来检测该对象是否已压缩，而不再次压缩文件。

配置 CloudFront 来压缩对象

要将 CloudFront 配置为压缩对象，请执行下面的所有 操作来更新您要对压缩对象执行的缓存行为：

1. 确保 Compress objects automatically（自动压缩对象）设置为 Yes（是）。（在 AWS CloudFormation 或 CloudFront API 中，将 `Compress` 设置为 `true`。）
2. 使用缓存策略指定缓存设置，并确保 Gzip 和 Botli 设置均已启用。（在 AWS CloudFormation 或 CloudFront API 中，将 `EnableAcceptEncodingGzip` 和 `EnableAcceptEncodingBrotli` 设置为 `true`。）
3. 确保缓存策略中的 TTL 值设置为大于零的值。当您将 TTL 值设置为零时，缓存处于禁用状态，CloudFront 不会压缩对象。

要更新缓存行为，可以使用以下任一工具：

- CloudFront 控制台
- AWS CloudFormation
- AWS 开发工具包和命令行工具

CloudFront 压缩的工作原理

当您将 CloudFront 配置为压缩对象时（请参阅上一部分），它的工作方式如下：

1. 查看器请求对象。查看器在请求中包含 `Accept-Encoding` HTTP 标头，标头值包含 `gzip`、`br` 或同时包含二者。这表示查看器支持压缩对象。当查看器同时支持 Gzip 和 Botli 时，CloudFront 将首选 Botli。

   > **Note**
   >
   > Chrome 和 Firefox Web 浏览器仅在使用 HTTPS 发送请求时支持 Brotli 压缩。这些浏览器不支持带 HTTP 请求的 Brotli。

2. 在边缘站点，CloudFront 将检查缓存中有无所请求对象的压缩副本。
3. 如果压缩的对象已在缓存中，则 CloudFront 将其发送到查看器并跳过剩余步骤。

   如果压缩的对象不在缓存中，则 CloudFront 将请求转发到源。

   > **Note**
   >
   > 如果缓存中已有对象的未压缩副本，CloudFront 可能会将其发送到查看器，而不将请求转发到源。例如，当 CloudFront 先前跳过了压缩时，可能会发生这种情况。发生这种情况时，CloudFront 会缓存未压缩的对象并继续处理该对象，直到对象过期、移出或失效。

4. 如果源返回了压缩对象（通过 HTTP 响应中存在的 `Content-Encoding` 标头指示），CloudFront 将该压缩对象发送给查看器，将其添加到缓存中，并跳过剩余步骤。CloudFront 不会再次压缩对象。

   如果源将未压缩的对象返回到 CloudFront（HTTP 响应中无 `Content-Encoding` 标头），则 CloudFront 将确定对象是否可压缩。有关 CloudFront 如何确定对象是否可压缩的更多信息，请参阅以下部分。

5. 如果对象可压缩，则 CloudFront 将压缩该对象，然后将其发送到查看器，并将其添加到缓存中。（在极少数情况下，CloudFront 可能会跳过压缩，然后将未压缩的对象发送给查看器。）

当 CloudFront 压缩对象时

以下列表提供了关于 CloudFront 何时压缩对象的详细信息。

请求使用 HTTP 1.0

如果对 CloudFront 的请求使用 HTTP 1.0，则 CloudFront 将删除 `Accept-Encoding` 标头，并且不会压缩响应中的对象。

`Accept-Encoding` 请求标头

如果 `Accept-Encoding` 标头在查看器请求中丢失，或者如果它不包含 `gzip` 或 `br` 作为值，则 CloudFront 不会压缩响应中的对象。如果 `Accept-Encoding` 标头包含其他值（例如 `deflate`），则 CloudFront 会先删除这些值，然后再将请求转发到源。

当 CloudFront 配置为压缩对象时，它自动在缓存键中和源请求中包含 `Accept-Encoding` 标头。

动态内容

CloudFront 并非总是压缩动态内容。有时对动态内容的响应会被压缩，有时不会。

当您将 CloudFront 配置为压缩对象时，内容已被缓存

CloudFront 在从源获取对象时压缩对象。当您将 CloudFront 配置为压缩对象时，CloudFront 不会压缩已在边缘站点中缓存的对象。此外，如果边缘站点上的缓存对象过期，并且 CloudFront 将该对象的另一个请求转发到源，则当源返回 HTTP 状态代码 304 时（这意味着，边缘站点中已有最新版本的对象），CloudFront 不会压缩该对象。如果您希望 CloudFront 压缩已在边缘站点中缓存的对象，需要使这些对象失效。有关更多信息，请参阅使文件失效以删除内容。

源已配置为压缩对象

如果将 CloudFront 配置为压缩对象，并且源也压缩了对象，则源中应包含 `Content-Encoding` 标头，向 CloudFront 指示该对象已经压缩。如果来自源的响应包含 `Content-Encoding` 标头，则无论标头值如何，CloudFront 都不会压缩对象。CloudFront 将响应发送给查看器，并在边缘站点缓存对象。

CloudFront 压缩的文件类型

有关 CloudFront 压缩的文件类型的完整列表，请参阅 CloudFront 压缩的文件类型。

CloudFront 压缩的对象大小

CloudFront 压缩大小介于 1000 字节和 10000000 字节之间的对象。

`Content-Length` 标头

源必须在响应中包含 `Content-Length` 标头，CloudFront 使用它来确定对象大小是否在 CloudFront 压缩的范围内。如果 `Content-Length` 标头丢失、包含无效值或包含超出 CloudFront 可压缩的大小范围的值，则 CloudFront 不会压缩该对象。

响应的 HTTP 状态代码

CloudFront 仅在响应的 HTTP 状态代码为 `200`、`403` 或 `404` 时压缩对象。

响应没有正文

当来自源的 HTTP 响应没有正文时，CloudFront 没有任何可以压缩的内容。

ETag 标头

CloudFront 有时会在压缩对象时修改 HTTP 响应中的 ETag 标头。有关更多信息，请参阅 the section called “ETag 标头转换”。

CloudFront 跳过压缩

CloudFront 将尽最大努力压缩对象。在极少数情况下，CloudFront 会跳过压缩。CloudFront 根据包括主机容量在内的各种因素做出此决定。如果 CloudFront 跳过对象的压缩，它会缓存未压缩的对象，并在该对象过期、被移出或失效之前继续将其提供给查看器。

CloudFront 压缩的文件类型

如果您将 CloudFront 配置为压缩对象，则 CloudFront 仅压缩 Content-Type 响应标头中具有以下值之一的对象：

- application/dash+xml
- application/eot
- application/font
- application/font-sfnt
- application/javascript
- application/json
- application/opentype
- application/otf
- application/pdf
- application/pkcs7-mime
- application/protobuf
- application/rss+xml
- application/truetype
- application/ttf
- application/vnd.apple.mpegurl
- application/vnd.mapbox-vector-tile

- application/vnd.ms-fontobject
- application/wasm
- application/xhtml+xml
- application/xml
- application/x-font-opentype
- application/x-font-truetype
- application/x-font-ttf
- application/x-httpd-cgi
- application/x-javascript
- application/x-mpegurl
- application/x-opentype
- application/x-otf
- application/x-perl
- application/x-ttf
- font/eot
- font/opentype
- font/otf
- font/ttf
- image/svg+xml
- text/css
- text/csv
- text/html
- text/javascript
- text/js
- text/plain
- text/richtext
- text/tab-separated-values
- text/xml
- text/x-component
- text/x-java-source

- text/x-script
- vnd.apple.mpegurl

ETag 标头转换

当来自源的未压缩对象包含有效的强 ETag HTTP 标头并且 CloudFront 压缩该对象时，CloudFront 还会将强 ETag 标头值转换为弱 ETag，并将弱 ETag 值返回到查看器。查看器可以存储弱 ETag 值并使用它来发送带 If-None-Match HTTP 标头的条件请求。这允许查看器、CloudFront 和源将对象的压缩版本和未压缩版本视为语义等效，从而减少不必要的数据传输。

有效的强 ETag 标头值以双引号字符 (") 开头。为了将强 ETag 值转换为弱值，CloudFront 会将字符 W/ 添加到强 ETag 值的开头。

当来自源的对象包含弱 ETag 标头值（以字符 W/ 开头的值）时，CloudFront 不会修改此值，并会在从源接收此值后将它返回到查看器。

当来自源的对象包含无效的 ETag 标头值（该值不以 " 或 W/ 开头）时，CloudFront 会删除 ETag 标头，并将对象返回到查看器而不带 ETag 响应标头。

有关更多信息，请参阅 MDN Web 文档中的以下页面：

- 指令（ETag HTTP 标头）
- 弱验证（HTTP 条件请求）
- If-None-Match HTTP 标头

使用 AWS WAF 保护功能

您可以使用 AWS WAF 来保护您的 CloudFront 分配和原始服务器，AWS WAF 是一个 Web 应用程序防火墙，可让您保护您的 Web 应用程序和 API，以在请求到达服务器之前阻止请求。有关更多详细信息，请参阅使用 CloudFront 和 AWS WAF 加速和保护您的网站。

要启用 AWS WAF 保护，您可以：

- 在 CloudFront 控制台中使用一键保护。一键保护将创建 AWS WAF Web 访问控制列表 (Web ACL)，配置规则以保护您的服务器免受常见的网络威胁，并将 Web ACL 附加到 CloudFront 分配。本节中的主题假设使用一键保护。
- 使用您在 AWS WAF 控制台中创建的预配置的 Web ACL（访问控制列表），或者使用 AWS WAF API。有关更多信息，请参阅《AWS WAF 开发人员指南》中的 Web 访问控制列表 (ACL)和《AWS WAF API 参考》中的 AssociateWebACL。

您可以在以下情况下启用 AWS WAF：

- 创建分配
- 使用安全控制面板编辑现有分配的安全设置

当您使用一键式保护功能时，CloudFront 会应用一组 AWS 推荐的保护措施，以便：

- 根据 Amazon 内部威胁情报，阻止 IP 地址受到潜在威胁。
- 防范 OWASP 前 10 名中描述的 Web 应用程序中最常见的漏洞。
- 防范恶意行为者发现应用程序漏洞。

> **Important**
>
> 如果要在 CloudFront 安全控制面板中查看安全指标，则必须启用 AWS WAF。如果未启用 AWS WAF，则只能使用安全控制面板来启用 AWS WAF 或配置 CloudFront 地理限制。有关控制面板的更多信息，请参阅本节后面部分中的在 CloudFront 安全控制面板中管理 AWS WAF 安全保护功能。

主题

为分配启用 AWS WAF

您可以在创建分配时启用 AWS WAF，也可以为现有的访问控制列表（ACL）启用安全保护功能。

如果您为 CloudFront 分配启用 AWS WAF，则还可以启用机器人控制功能并按机器人类别配置安全保护功能。

主题

为新分配启用 AWS WAF

以下步骤演示如何在创建新 CloudFront 分配时启用 AWS WAF。

为新分配启用 AWS WAF

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 在导航窗格中，选择分配，然后选择创建分配。
3. 根据需要，按照创建分配中的步骤进行操作。
4. 在 Web 应用程序防火墙部分，选择编辑，然后选择启用安全保护。
5. 填写以下字段：

    - 使用监控模式 – 如果要先收集数据以测试保护的工作原理，可以启用监控模式。启用监控模式后，如果保护处于活动状态，则不会阻止请求。相反，监控模式会收集有关在保护处于活动状态时将被阻止的请求的数据。当您准备好开始阻止时，可以在安全页面上启用阻止功能。
    - 其他保护 – 选择要启用的任何选项。如果您启用了速率限制，请参阅 the section called “设置速率限制”以了解更多信息。

- 价格估算 – 您可以打开该部分以显示一个字段，在该字段中输入不同的请求数/月份，并查看新的估算值。

6. 查看其余分配设置，然后选择创建分配。

当您创建分配时，CloudFront 会创建一个安全控制面板。您可以使用此控制面板禁用或启用 AWS WAF。如果您尚未启用 AWS WAF，控制面板中的图表和图形将保持空白。

使用现有 Web ACL

如果您已有 Web ACL，则可以使用它来代替 AWS WAF 提供的保护功能。

使用现有 AWS WAF 配置

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 请执行以下操作之一：
   a. 选择创建分配并按照创建分配中的步骤操作，然后返回本主题。
   b. 选择现有配置，然后选择安全选项卡。
3. 在 Web 应用程序防火墙（WAF）部分，选择编辑，然后选择启用安全保护。
4. 选择使用现有 WAF 配置。此选项仅在配置了 Web ACL 时才会出现。
5. 从选择 Web ACL 表中选择现有的 Web ACL。
6. 查看其余分配设置，然后选择创建分配。

启用机器人控制功能

如果您为 CloudFront 分配启用 AWS WAF，则可以在 CloudFront 控制台的安全控制面板下，查看给定时间范围内的机器人请求。您也可在此处启用或禁用机器人控制功能。

启用机器人控制功能后会产生费用。安全控制面板提供了成本估算。

如果您启用机器人控制功能，则安全控制面板会按每种机器人类型和类别显示机器人的流量。如果您禁用机器人控制功能，则会根据请求采样显示机器人的流量。

启用机器人控制功能

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台

2. 在导航窗格中，选择分配，然后选择要更改的分配。
3. 选择安全性选项卡。
4. 向下滚动到给定时间范围内的机器人请求部分，然后选择启用机器人控制功能。
5. 在机器人控制功能对话框的配置下，选中为普通机器人启用机器人控制功能复选框。
6. 选择保存更改。

按机器人类别配置保护功能

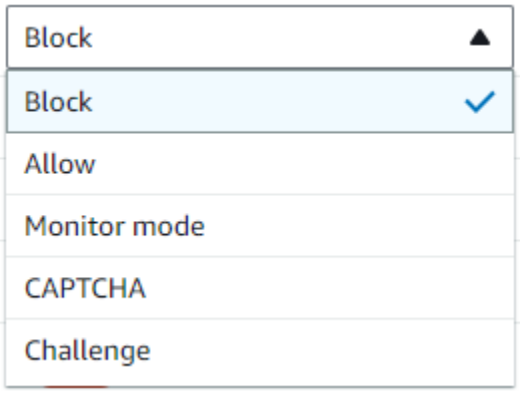
启用机器人控制功能后，您可以配置如何按机器人类别处理每个未经验证的机器人。例如，您可以将 HTTP 库机器人设置为监控模式，并将质询分配给链接检查工具。

> **Note**
>
> AWS 已知的常见且可验证的机器人（如已知的搜索引擎爬网程序）不受您在此设置的操作的限制。机器人控制功能会确认经过验证的机器人来自他们声称的来源，然后再将其标记为已验证。

为类别配置保护功能

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 在导航窗格中，选择分配，然后选择要更改的分配。
3. 选择安全性选项卡。
4. 在按机器人类别划分的请求图表中，指向未验证的机器人操作列中的任何项目，然后选择编辑图标。

5. 打开结果列表并选择下列选项之一：
   - 阻止
   - 允许
   - 监控模式
   - 验证码
   - 质询



6. 选中列表旁边的复选标记以提交您的更改。

在 CloudFront 安全控制面板中管理 AWS WAF 安全保护功能

CloudFront 会为您的每个分配创建一个安全控制面板。使用 CloudFront 控制台中的控制面板。借助控制面板，您可以在单一位置结合使用 CloudFront 和 AWS WAF，来监控和管理对 Web 应用程序的一般安全保护。控制面板提供以下任务和数据：

- 安全配置 – 您可以启用和禁用 AWS WAF 保护功能，并查看任何特定于应用程序的保护功能，例如 WordPress 保护。
- 安全趋势 – 其中包括允许和阻止的请求、质询和验证码请求以及主要攻击类型。您可以看到流量比率以及它们随时间推移的变化。例如，如果所有请求都增加了 3%，但允许的请求增加了 14%，则意味着您在当前时段允许了更大一部分流量通过。
- 机器人请求 – 您可以查看有多少流量来自机器人、有哪些类型的机器人（已验证与未验证），以及机器人类型（已验证与未验证）的百分比分配如何随时间变化。有关启用机器人控制功能的更多信息，请参阅启用机器人控制功能。
- 请求日志 – 日志数据可以帮助解答有关安全趋势或机器人请求的问题。您无需编写查询即可搜索日志，还可以查看汇总图表，以帮助确定筛选后的日志集是否主要由 HTTP 方法、IP 地址、URI 路径或国家/地区的子集驱动。您可以将鼠标悬停在图表中的值上，并阻止 IP 地址和国家/地区。有关更多信息，请参阅 启用 AWS WAF 日志。
- 地理限制管理 – CloudFront 和 AWS WAF 提供地理限制功能。CloudFront 免费提供地理限制功能，但安全控制面板中不会显示 CloudFront 地理限制功能的指标。要查看被屏蔽的国家/地区请求的请求

指标，您必须使用 AWS WAF 地理限制功能。为此，请将鼠标悬停在安全控制面板中的国家/地区栏上，然后屏蔽该国家/地区。有关更多信息，请参阅 使用 CloudFront 地理限制。

- 如果您之前在 CloudFront 控制台之外创建了用于阻止国家/地区的自定义 AWS WAF 规则，则阻止选项可能不可用。

主题

先决条件

如果要在 CloudFront 安全控制面板中查看安全指标，则必须启用 AWS WAF。如果未启用 AWS WAF，则只能使用安全控制面板来启用 AWS WAF 或配置 CloudFront 地理限制。

有关启用 AWS WAF 的更多信息，请参阅 为分配启用 AWS WAF。

启用 AWS WAF 日志

AWS WAF 日志数据可以帮助您隔离特定的流量模式。例如，日志可以向您显示某些流量的来源或用途。

如果您启用对 CloudWatch 的 AWS WAF 日志记录，则 CloudFront 安全控制面板会查询、汇总和显示来自 CloudWatch 日志的见解。我们不收取安全控制面板的使用费用，但是 CloudWatch 定价适用于通过控制面板查询的日志。有关更多信息，请参阅 Amazon CloudWatch 定价。

启用日志

1. 在请求数/月框中输入您的预期请求量，以估算启用日志的成本。
2. 选中启用 AWS WAF 日志复选框。
3. 请选择 启用。

CloudFront 会创建一个 CloudWatch 日志组并更新您的 AWS WAF 配置以开始将日志记录到 CloudWatch。首次使用时，日志数据可能需要几分钟才能显示。图表的请求部分列出了每个请求。在单个请求下方，条形图按 HTTP 方法、主要 URI 路径、主要 IP 地址和主要国家/地区汇总数据。图表可以帮助您找到模式。例如，您可能会看到来自单个 IP 地址的请求量不成比例，或者看到来自您以前在日志中未见过的国家/地区的数据。您可以根据国家/地区、主机标头和其他属性筛选请求，以帮助查

找不需要的流量。识别出那些流量后，将鼠标指针悬停在单个请求或图表项上，然后阻止某个 IP 地址或国家/地区。

> **Note**
>
> 显示的指标基于 Web ACL。因此，如果您将同一个 Web ACL 关联到多个分配，您将看到 Web ACL 的所有指标，而不仅仅是针对该分配处理的 AWS WAF 请求。

设置速率限制

速率限制是您在配置安全保护时可能会收到的建议之一。

CloudFront 始终在监控模式下启用速率限制。启用监控模式后，CloudFront 会捕获一些指标，告诉您是否已超过您在速率限制字段中配置的速率、超出频率以及超出幅度。

保存分配后，CloudFront 开始根据速率限制字段中的数字收集数据。

您可以在任何 CloudFront 分配的安全选项卡上的安全 – Web 应用程序防火墙 (WAF) 部分中管理速率限制设置。

设置速率限制

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 在导航窗格中，选择分配，然后选择要更改的分配。
3. 选择安全性选项卡。
4. 在 Web 应用程序防火墙（WAF）部分的速率限制旁边，选择监控模式消息，以显示包含所收集数据的详细信息的对话框。您可以选择更改速率限制。微调速率后，可以选择启用阻止（在该对话框中）以停用监视模式。CloudFront 将开始阻止超过指定速率限制的请求。

启用 AWS WAF 安全保护功能

如果您的分配不需要 AWS WAF 安全保护，则可以使用 CloudFront 控制台禁用此功能。

如果您之前启用了 AWS WAF 保护，但没有选择现有 WAF 配置（也称为一键保护），CloudFront 会自动为您创建一个 Web ACL。对于以这种方式创建的 Web ACL，CloudFront 控制台将取消资源关联并删除该 Web ACL。

取消关联 Web ACL 与将其删除不同。取消关联会从您的分配中移除 Web ACL，但不会将其从您的 AWS 账户中删除。有关更多信息，请参阅《AWS WAF、AWS Firewall Manager 和 AWS Shield Advanced 开发人员指南》中的将 Web ACL 与 AWS 资源关联或取消关联。

要禁用 AWS WAF 保护并取消 Web ACL 与您的分配的关联，请参阅以下步骤。

在 CloudFront 中禁用 AWS WAF 安全保护

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 在导航窗格中，选择分配，然后选择要更改的分配。
3. 选择安全选项卡，然后选择编辑。
4. 在 Web 应用程序防火墙（WAF）部分，选择禁用 AWS WAF 保护。
5. 选择保存更改。

> **注意**
>
> - 如果您禁用了 AWS WAF 安全保护，但仍想从 AWS 账户中删除 Web ACL，则可以手动将其删除。按照步骤删除 Web ACL。在 AWS WAF 和 Shield 控制台中，对于 Web ACL 页面，您必须选择全局（CloudFront）列表才能查找 Web ACL。
> - 当您从 CloudFront 控制台删除分配时，如果选择一键保护，CloudFront 还会尝试删除该 Web ACL。此尝试是尽力而为，不能保证成功删除。有关更多信息，请参阅 删除分配。

配置安全访问和限制对内容的访问

CloudFront 在保护其传输的内容方面提供了多种选择。可使用 CloudFront 通过以下方式来保护和限制对内容的访问：

- 配置 HTTPS 连接
- 阻止特定地理位置的用户访问内容
- 要求用户使用 CloudFront 签名 URL 或签名 Cookie 来访问内容
- 为特定的内容字段设置字段级加密
- 使用 AWS WAF 控制对您的内容的访问

主题

将 HTTPS 与 CloudFront 结合使用

您可以将 CloudFront 配置为要求查看器使用 HTTPS，以便在 CloudFront 与查看器通信时加密连接。您也可以将 CloudFront 配置为通过源使用 HTTPS，以便在 CloudFront 与源通信时加密连接。

如果您将 CloudFront 配置为要求使用 HTTPS 与查看器和您的源进行通信，当 CloudFront 接收请求时将发生以下操作。

1. 查看器向 CloudFront 提交 HTTPS 请求。查看器和 CloudFront 之间存在一些 SSL/TLS 协商。最后，查看器以加密格式提交请求。
2. 如果 CloudFront 边缘站点包含缓存响应，CloudFront 将为响应加密并将其返回给查看器，然后查看器对其进行解密。
3. 如果 CloudFront 边缘站点不包含缓存响应中，CloudFront 则与您的源执行 SSL/TLS 协商，当协商完成时，将请求以加密格式转发给您的源。

4. 您的源会对请求进行解密、处理（生成响应），加密响应，并将响应返回给 CloudFront。
5. CloudFront 对响应进行解密，并对其重新加密，然后将其转发给查看器。CloudFront 还将缓存边缘站点中的响应，以便在下次请求时可用。
6. 查看器对响应进行解密。

无论您的源是 Amazon S3 存储桶、MediaStore 还是 HTTP/S 服务器之类的自定义源，过程基本相同：

> **Note**
>
> 为了帮助阻止 SSL 重新协商类型的攻击，CloudFront 不支持查看方和源请求的重新协商。

有关在查看器和 CloudFront 之间以及 CloudFront 和源之间如何使用 HTTPS 的信息，请参阅以下主题。

主题

要求在查看器和 CloudFront 之间使用 HTTPS 进行通信

您可以在 CloudFront 分配中配置一个或多个缓存行为，以便要求在查看器和 CloudFront 之间使用 HTTPS 进行通信。您也可以配置一个或多个缓存行为以允许 HTTP 和 HTTPS，以便 CloudFront 对某些对象要求使用 HTTPS，而对其他对象不做这一要求。配置步骤取决于您在对象 URL 中使用的域名：

- 如果您使用 CloudFront 指派给您的域名（例如 d111111abcdef8.cloudfront.net），您可以针对要求 HTTPS 通信的一个或多个缓存行为更改 查看器协议策略设置。在该配置中，CloudFront 提供 SSL/TLS 证书。

  要使用 CloudFront 控制台更改查看器协议策略值，请参阅本节后面的步骤。

有关如何使用 CloudFront API 更改 `ViewerProtocolPolicy` 元素值的信息，请参阅《Amazon CloudFront API 参考》中的 UpdateDistribution。

- 如果您使用自己的域名（例如 example.com），则需要更改若干 CloudFront 设置。您还需要使用 AWS Certificate Manager (ACM) 提供的 SSL/TLS 证书，或将证书从第三方证书颁发机构导入到 ACM 或 IAM 证书存储。有关更多信息，请参阅 使用备用域名和 HTTPS。

> **Note**
>
> 如果您希望确保在 CloudFront 从源获取对象并且查看器从 CloudFront 获取这些对象时，对象得到了加密，请始终在 CloudFront 和您的源之间使用 HTTPS。如果您最近在 CloudFront 和您的源之间从 HTTP 更改为 HTTPS，建议您使 CloudFront 边缘站点中的对象失效。CloudFront 将对象返回给查看器，而无论查看器使用的协议（HTTP 或 HTTPS）与 CloudFront 用于获取对象的协议是否匹配。有关删除或替换分配中的对象的更多信息，请参阅 添加、删除或替换 CloudFront 分配的内容。

要求查看器使用 HTTPS

如果针对一个或多个缓存行为要求在查看器和 CloudFront 之间使用 HTTPS，请执行以下步骤。

将 CloudFront 配置为要求在查看器与 CloudFront 之间使用 HTTPS

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在 CloudFront 控制台的顶部窗格中，选择您要更新的分配的 ID。
3. 在行为选项卡中，选择要更新的缓存行为，然后选择编辑。
4. 为查看器协议策略指定以下值之一：

   Redirect HTTP to HTTPS

   查看器可以使用两种协议。HTTP `GET` 和 `HEAD` 请求会自动重定向到 HTTPS 请求。CloudFront 返回 HTTP 状态代码 301（永久移动）以及新的 HTTPS URL。然后，查看器会使用此 HTTPS URL 将请求重新提交到 CloudFront。

> **Important**
>
> 如果您通过 HTTP 使用 HTTP 到 HTTPS 的缓存行为以及请求协议版本 HTTP 1.1 或更高版本发送 POST、PUT、DELETE、OPTIONS 或 PATCH，CloudFront 会使用 HTTP 状态代码 307（临时重定向）将请求重定向到 HTTPS 位置。这可确保使用相同的方法和正文负载将请求再次发送到新位置。
>
> 如果您使用低于 HTTP 1.1 的请求协议版本通过 HTTP 到 HTTPS 的缓存行为发送 POST、PUT、DELETE、OPTIONS 或 PATCH 请求，CloudFront 将返回 HTTP 状态代码 403（禁止访问）。

在查看器发出将重定向到 HTTPS 请求的 HTTP 请求时，会产生针对这两个请求的 CloudFront 费用。对于 HTTP 请求，仅对该请求和 CloudFront 返回到查看器的标头计费。对于 HTTPS 请求，对该请求、标头和由源返回的对象计费。

仅 HTTPS

查看器只有使用 HTTPS 才能访问您的内容。如果查看器发送 HTTP 请求而不是 HTTPS 请求，则 CloudFront 将返回 HTTP 状态代码 403（禁止访问）且不会返回对象。

5. 选择保存更改。
6. 针对要求在查看器和 CloudFront 之间使用 HTTPS 的其他每个缓存行为，重复步骤 3 到 5。
7. 请确认以下内容，然后在生产环境中使用更新后的配置：
   - 每个缓存行为中的路径模式仅适用于您希望查看器使用 HTTPS 的请求。
   - 缓存行为按您希望 CloudFront 评估它们的顺序列出。有关更多信息，请参阅 路径模式。
   - 缓存行为将请求路由到正确的源。

要求在 CloudFront 与您的自定义源之间使用 HTTPS 进行通信

您可以要求使用 HTTPS 在 CloudFront 与您的源之间的通信

> **Note**
>
> 如果您的源是配置为网站终端节点的 Amazon S3 存储桶，则无法将 CloudFront 配置为将 HTTPS 与源结合使用，因为 Amazon S3 不支持对网站终端节点使用 HTTPS。

如需要求在 CloudFront 和您的源之间的需要使用 HTTPS，请按照本主题中的过程执行以下操作：

1. 在您的分配中，更改源的 Origin Protocol Policy (源协议策略) 设置
2. 在您的自定义源服务器上安装 SSL/TLS 证书（当您使用 Amazon S3 源或某些其他 AWS 源时，不需要执行此操作）。

主题

要求自定义源使用 HTTPS

以下过程介绍了如何配置 CloudFront 以使用 HTTPS 与 Elastic Load Balancing 负载均衡器、Amazon EC2 实例或其他自定义源进行通信。有关使用 CloudFront API 更新分配的信息，请参阅《Amazon CloudFront API 参考》中的 UpdateDistribution。

将 CloudFront 配置为要求在 CloudFront 和您的自定义源之间使用 HTTPS

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在 CloudFront 控制台的顶部窗格中，选择您要更新的分配的 ID。
3. 在行为选项卡中，选择要更新的源，然后选择编辑。
4. 更新以下设置：

   源协议策略

   为您的分配中的适用源更改源协议策略：

   - 仅 HTTPS – CloudFront 仅使用 HTTPS 与自定义源进行通信。
   - 匹配查看器 – CloudFront 使用 HTTP 或 HTTPS 与自定义源进行通信，具体取决于查看器请求协议。例如，如果您源协议策略选择匹配查看器，并且查看器使用 HTTPS 从 CloudFront 请求对象，则 CloudFront 也会使用 HTTPS 将请求转发给您的源。

     只有在为查看器协议策略指定将 HTTP 重定向到 HTTPS 或仅 HTTPS 时，才能选择匹配查看器。

     请注意，CloudFront 仅缓存对象一次，即使查看器使用 HTTP 和 HTTPS 协议发出请求也是如此。

Origin SSL Protocols

为您的分配中的适用源选择 Origin SSL Protocols。由于 SSLv3 协议的安全性较低，因此，建议您仅在源不支持 TLSv1 或更高版本的情况下选择 SSLv3。TLSv1 握手与 SSLv3 向后和向前兼容，但 TLSv1.1 及更高版本不是这样。当您选择 SSLv3 时，CloudFront 仅 发送 SSLv3 握手请求。

5. 选择保存更改。
6. 针对要求在 CloudFront 和您的自定义源之间使用 HTTPS 的其他每个源，重复步骤 3 到 5。
7. 请确认以下内容，然后在生产环境中使用更新后的配置：

   - 每个缓存行为中的路径模式仅适用于您希望查看器使用 HTTPS 的请求。
   - 缓存行为按您希望 CloudFront 评估它们的顺序列出。有关更多信息，请参阅 路径模式。
   - 缓存行为将请求路由到更改了源协议策略的源。

在自定义源上安装 SSL/TLS 证书

您可以在自定义源上使用来自以下来源的 SSL/TLS 证书：

- 如果您的源是 Elastic Load Balancing 负载均衡器，则可以使用 AWS Certificate Manager (ACM) 提供的证书。您也可以使用信任的第三方证书颁发机构签署并导入 ACM 的证书。
- 对于 Elastic Load Balancing 负载均衡器之外的源，您必须使用由信任的第三方证书颁发机构 (CA)（例如，Comodo、DigiCert 或 Symantec）签署的证书。

从源返回的证书必须包括以下域名之一：

- 源的源域字段中的域名（CloudFront API 中的 `DomainName` 字段）。
- `Host` 标头中的域名，如果缓存行为配置为将 `Host` 标头转发至源。

当 CloudFront 使用 HTTPS 与您的源进行通信时，CloudFront 会验证信任的证书颁发机构颁发的证书。CloudFront 与 Mozilla 支持相同的证书颁发机构。有关当前列表，请参阅 Mozilla 包含的 CA 证书列表。您无法使用自签名证书在 CloudFront 和您的源之间进行 HTTPS 通信。

> **Important**
>
> 如果源服务器返回过期证书、无效证书或自签名证书，或者如果源服务器以错误顺序返回证书链，CloudFront 将中断 TCP 连接，向查看器返回 HTTP 错误代码 502（无效网关），并将 X-

`Cache` 标头设置为 `Error from cloudfront`。此外，如果不存在完整的证书链（包括中间证书），则 CloudFront 将中断 TCP 连接。

要求在 CloudFront 与 Amazon S3 源之间使用 HTTPS 进行通信

当您的源是 Amazon S3 存储桶时，用于使用 HTTPS 以与 CloudFront 进行通信的选项取决于您使用存储桶的方式。如果您的 Amazon S3 存储桶配置为网站终端节点，则您无法将 CloudFront 配置为使用 HTTPS 与您的源进行通信，因为 Amazon S3 不支持该配置中的 HTTPS 连接。

当您的源是支持 HTTPS 通信的 Amazon S3 存储桶时，CloudFront 始终使用查看器用来提交请求的协议将请求转发到 S3。协议（仅自定义源）的默认设置为匹配查看器，并且不能更改。

如果您希望要求在 CloudFront 和 Amazon S3 之间使用 HTTPS 进行通信，则必须将查看器协议策略的值更改为将 HTTP 重定向到 HTTPS 或仅 HTTPS。有关如何使用 CloudFront 控制台更改查看器协议策略的信息，请参阅本节后面的步骤。有关使用 CloudFront API 更新分配的 `ViewerProtocolPolicy` 元素的信息，请参阅 Amazon CloudFront API 参考中的 UpdateDistribution。

当您将 HTTPS 与支持 HTTPS 通信的 Amazon S3 存储桶结合使用时，Amazon S3 会提供 SSL/TLS 证书，因此您无需提供。

要求 Amazon S3 源使用 HTTPS

以下步骤演示如何将 CloudFront 配置为要求通过 HTTPS 连接到您的 Amazon S3 源。

将 CloudFront 配置为要求通过 HTTPS 连接到您的 Amazon S3 源

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在 CloudFront 控制台的顶部窗格中，选择您要更新的分配的 ID。
3. 在 Behaviors 选项卡中，选择要更新的缓存行为，然后选择 Edit。
4. 为查看器协议策略指定以下值之一：

   将 HTTP 重定向到 HTTPS

   查看器可使用两种协议，但 HTTP 请求将自动重定向到 HTTPS 请求。CloudFront 返回 HTTP 状态代码 301（永久移动）以及新的 HTTPS URL。然后，查看器会使用此 HTTPS URL 将请求重新提交到 CloudFront。

> **Important**
>
> CloudFront 不将 DELETE、OPTIONS、PATCH、POST 或 PUT 请求从 HTTP 重定向到 HTTPS。如果您将一个缓存行为配置为重定向到 HTTPS，CloudFront 会针对该缓存行为的 HTTP DELETE、OPTIONS、PATCH、POST 或 PUT 请求响应 HTTP 状态代码 403（禁止访问）。

在查看器发出将重定向到 HTTPS 请求的 HTTP 请求时，会产生针对这两个请求的 CloudFront 费用。对于 HTTP 请求，仅对该请求和 CloudFront 返回到查看器的标头计费。对于 HTTPS 请求，对该请求、标头和由源返回的对象计费。

HTTPS Only

查看器只有使用 HTTPS 才能访问您的内容。如果查看器发送 HTTP 请求而不是 HTTPS 请求，则 CloudFront 将返回 HTTP 状态代码 403（禁止访问）且不会返回对象。

5. 选择是，编辑。
6. 针对要求在查看器和 CloudFront 之间以及 CloudFront 和 S3 之间使用 HTTPS 的其他每个缓存行为，重复步骤 3 到 5。
7. 请确认以下内容，然后在生产环境中使用更新后的配置：
   - 每个缓存行为中的路径模式仅适用于您希望查看器使用 HTTPS 的请求。
   - 缓存行为按您希望 CloudFront 评估它们的顺序列出。有关更多信息，请参阅 路径模式。
   - 缓存行为将请求路由到正确的源。

查看器和 CloudFront 之间支持的协议和密码

当您需要在查看器和 CloudFront 分配之间使用 HTTPS 时，必须选择一项安全策略来确定以下设置。

- CloudFront 与查看器通信时使用的最低 SSL/TLS 协议。
- CloudFront 可用于加密与查看器之间的通信的密码。

要选择安全策略，请为 安全策略（最低 SSL/TLS 版本） 指定合适的值。下表列出了每个安全策略中，CloudFront 可用的协议和密码。

查看器至少必须支持这些受支持的密码中的一个，才能与 CloudFront 建立 HTTPS 连接。CloudFront 按列出的顺序从查看器支持的密码中选择一种密码。另请参阅 OpenSSL、s2n 和 RFC 密码名称。

	安全策略						
	SSLv3	TLSv1	TLSv1_26	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
支持的 SSL/TLS 协议							
TLSv1.3	◆	◆	◆	◆	◆	◆	◆
TLSv1.2	◆	◆	◆	◆	◆	◆	◆
TLSv1.1	◆	◆	◆	◆			
TLSv1	◆	◆	◆				
SSLv3	◆						
支持的 TLSv1.3 密码							
TLS_AES_128_GCM_SHA256	◆	◆	◆	◆	◆	◆	◆
TLS_AES_256_GCM_SHA384	◆	◆	◆	◆	◆	◆	◆
TLS_CHACHA20_POLY1305_SHA256	◆	◆	◆	◆	◆	◆	◆
支持的 ECDSA 密码							
ECDHE-ECDSA-AES128-GCM-SHA256	◆	◆	◆	◆	◆	◆	◆
ECDHE-ECDSA-AES128-SHA256	◆	◆	◆	◆	◆	◆	
ECDHE-ECDSA-AES128-SHA	◆	◆	◆	◆			

	安全策略						
	SSLv3	TLSv1	TLSv1_26	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
ECDHE-ECDSA-AES256-GCM-SHA384	♦	♦	♦	♦	♦	♦	♦
ECDHE-ECDSA-CHACHA20-POLY1305	♦	♦	♦	♦	♦	♦	♦
ECDHE-ECDSA-AES256-SHA384	♦	♦	♦	♦	♦	♦	
ECDHE-ECDSA-AES256-SHA	♦	♦	♦	♦			
支持的 RSA 密码							
ECDHE-RSA-AES128-GCM-SHA256	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦	♦	♦	♦	♦	
ECDHE-RSA-AES128-SHA	♦	♦	♦	♦			
ECDHE-RSA-AES256-GCM-SHA384	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-CHACHA20-POLY1305	♦	♦	♦	♦	♦	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦	♦	♦	♦	♦	
ECDHE-RSA-AES256-SHA	♦	♦	♦	♦			

	安全策略						
	SSLv3	TLSv1	TLSv1_26	TLSv1.1_016	TLSv1.2_018	TLSv1.2_019	TLSv1.2_2021
AES128-GCM-SHA256	♦	♦	♦	♦	♦		
AES256-GCM-SHA384	♦	♦	♦	♦	♦		
AES128-SHA256	♦	♦	♦	♦	♦		
AES256-SHA	♦	♦	♦	♦			
AES128-SHA	♦	♦	♦	♦			
DES-CBC3-SHA	♦	♦					
RC4-MD5	♦						

OpenSSL、s2n 和 RFC 密码名称

OpenSSL 和 s2n 使用的密码名称与 TLS 标准使用的不同（RFC 2246、RFC 4346、RFC 5246 和 RFC 8446）。下表为每个密码列出了 OpenSSL 和 s2n 名称及对应的 RFC 名称。

对于使用椭圆曲线密钥交换算法的密码，CloudFront 支持以下椭圆曲线：

- prime256v1
- secp384r1
- X25519

OpenSSL 和 s2n 密码名称	RFC 密码名称
支持的 TLSv1.3 密码	
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256

OpenSSL 和 s2n 密码名称	RFC 密码名称
支持的 ECDSA 密码	
ECDHE-ECDSA-AES128- GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE-ECDSA-AES128-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE-ECDSA-AES128-SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
ECDHE-ECDSA-AES256- GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE-ECDSA-CHACHA20-POLY1305	TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
ECDHE-ECDSA-AES256-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
ECDHE-ECDSA-AES256-SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
支持的 RSA 密码	
ECDHE-RSA-AES128- GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256- GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

OpenSSL 和 s2n 密码名称	RFC 密码名称
ECDHE-RSA-CHACHA20-POLY1305	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

查看器和 CloudFront 之间受支持的签名方案

CloudFront 支持以下用于查看器和 CloudFront 之间的连接的签名方案。

- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PSS_PSS_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PSS_RSAE_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA512

- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SHA384
- TLS_SIGNATURE_SCHEME_ECDSA_SHA512
- TLS_SIGNATURE_SCHEME_ECDSA_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SECP256R1_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SECP384R1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA1
- TLS_SIGNATURE_SCHEME_ECDSA_SHA1

CloudFront 与源之间受支持的协议和密码

如果您选择要求在 CloudFront 和源之间使用 HTTPS，则可以确定安全连接允许使用哪项 SSL/TLS 协议，并且 CloudFront 可以使用下表中列出的任何 ECDSA 或 RSA 密码连接到源。您的源至少必须支持这些密码中的其中一个，才能使 CloudFront 与您的源建立 HTTPS 连接。

OpenSSL 和 s2n 使用的密码名称与 TLS 标准使用的不同（RFC 2246、RFC 4346、RFC 5246 和 RFC 8446）。下表为每个密码列出了 OpenSSL 和 s2n 名称及对应的 RFC 名称。

对于使用椭圆曲线密钥交换算法的密码，CloudFront 支持以下椭圆曲线：

- prime256v1
- secp384r1
- X25519

OpenSSL 和 s2n 密码名称	RFC 密码名称
支持的 ECDSA 密码	
ECDHE-ECDSA-AES256- GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
ECDHE-ECDSA-AES256-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

OpenSSL 和 s2n 密码名称	RFC 密码名称
ECDHE-ECDSA-AES256-SHA	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
ECDHE-ECDSA-AES128- GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
ECDHE-ECDSA-AES128-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
ECDHE-ECDSA-AES128-SHA	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
支持的 RSA 密码	
ECDHE-RSA-AES256- GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
ECDHE-RSA-AES128- GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA

OpenSSL 和 s2n 密码名称	RFC 密码名称
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

CloudFront 与源之间受支持的签名方案

CloudFront 支持以下签名方案，用于 CloudFront 和源之间的连接。

- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA256
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA384
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA512
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA224
- TLS_SIGNATURE_SCHEME_ECDSA_SHA256
- TLS_SIGNATURE_SCHEME_ECDSA_SHA384
- TLS_SIGNATURE_SCHEME_ECDSA_SHA512
- TLS_SIGNATURE_SCHEME_ECDSA_SHA224
- TLS_SIGNATURE_SCHEME_RSA_PKCS1_SHA1
- TLS_SIGNATURE_SCHEME_ECDSA_SHA1

使用备用域名和 HTTPS

如果您希望在文件的 URL 中使用您自己的域名（例如 `https://www.example.com/image.jpg`），并希望您的查看器使用 HTTPS，则必须完成以下主题中的步骤。（如果您在 URL 中使用默认 CloudFront 分配域名，例如 `https://d111111abcdef8.cloudfront.net/image.jpg`，请按照以下主题中的指导进行操作：要求在查看器和 CloudFront 之间使用 HTTPS 进行通信。）

> **Important**
>
> 将某证书添加到您的分配时，CloudFront 会立即将该证书传播到其所有边缘站点。当新的边缘站点可用时，CloudFront 也会将该证书传播到这些位置。您不能限制 CloudFront 将证书传播到的边缘站点。

主题

选择 CloudFront 如何处理 HTTPS 请求

如果希望您的查看器使用 HTTPS 并对您的文件使用备用域名，请为 CloudFront 处理 HTTPS 请求的方式选择下列选项之一：

- 使用服务器名称指示 (SNI) – 建议
- 在每个边缘站点使用专用 IP 地址

本部分说明了每个选项的工作方式。

使用 SNI 处理 HTTPS 请求（适用于大多数客户端）

服务器名称指示 (SNI) 是对 TLS 协议的扩展，2010 年以后发布的浏览器和客户端均支持。如果您将 CloudFront 配置为使用 SNI 处理 HTTPS 请求，则 CloudFront 将备用域名与每个边缘站点中的 IP 地址关联。当查看器提交针对内容的 HTTPS 请求时，DNS 将该请求传送到正确边缘站点的 IP 地址。指向您域名的 IP 地址在 SSL/TLS 握手协商期间确定；IP 地址并非专用于您的分发。

在建立 HTTPS 连接的过程的早期便进行了 SSL/TLS 协商。如果 CloudFront 无法立即确定该请求所针对的域，则它会中断连接。在支持 SNI 的查看器针对您的内容提交 HTTPS 请求时，将发生以下操作：

1. 查看器自动从请求 URL 中获取域名，并将其添加到 TLS 客户端问候消息的 SNI 扩展中。
2. 当 CloudFront 收到 TLS 客户端问候时，它会使用 SNI 扩展中的域名来查找匹配的 CloudFront 分配并发回关联的 TLS 证书。
3. 查看器和 CloudFront 执行 SSL/TLS 协商。

4. CloudFront 将请求的内容返回到查看器。

有关支持 SNI 的浏览器的当前列表，请参阅 Wikipedia 条目服务器名称指示。

如果您希望使用 SNI，但您的某些用户的浏览器不支持 SNI，则您有以下几种选择：

- 配置 CloudFront 以使用专用 IP 地址而不是 SNI 来提供 HTTPS 请求。有关更多信息，请参阅 使用专用 IP 地址处理 HTTPS 请求（适用于所有客户端）。
- 使用 CloudFront SSL/TLS 证书而不是自定义证书。这要求您在文件的 URL（例如 `https://d111111abcdef8.cloudfront.net/logo.png`）中为您的分配使用 CloudFront 域名。

  如果您使用默认 CloudFront 证书，查看器必须支持 SSL 协议 TLSv1 或更高版本。CloudFront 不支持 SSLv3 使用默认的 CloudFront 证书。

  您还必须将 CloudFront 使用的 SSL/TLS 证书从自定义证书更改为默认 CloudFront 证书：

  - 如果您尚未使用分配来分发内容，则只需更改配置。有关更多信息，请参阅 更新分配。
  - 如果您已使用分配来分发内容，则必须创建新的 CloudFront 分配并更改文件的 URL，以便减少或消除内容不可用的时间。有关更多信息，请参阅 从自定义 SSL/TLS 证书恢复为默认 CloudFront 证书。
- 如果您可以控制用户使用的浏览器，请让用户将浏览器升级为支持 SNI 的浏览器。
- 使用 HTTP 而不是 HTTPS。

使用专用 IP 地址处理 HTTPS 请求（适用于所有客户端）

服务器名称指示 (SNI) 是一种将请求与域关联的方法。另一种方法是使用专用 IP 地址。如果您的用户无法升级到 2010 年之后发布的浏览器或客户端，您可以使用专用 IP 地址为 HTTPS 请求提供服务。有关支持 SNI 的浏览器的当前列表，请参阅 Wikipedia 条目服务器名称指示。

> **Important**
>
> 如果您将 CloudFront 配置为使用专用 IP 地址提供 HTTPS 请求，则每月会产生额外的费用。一旦您将 SSL/TLS 证书与某个分配关联并启用该分配，就将开始计费。有关 CloudFront 定价的更多信息，请参阅 Amazon CloudFront 定价。另请参阅Using the Same Certificate for Multiple CloudFront Distributions。

当您将 CloudFront 配置为使用专用 IP 地址处理 HTTPS 请求时，CloudFront 会将您的证书与每个 CloudFront 边缘站点中的专用 IP 地址关联。在查看器针对您的内容提交 HTTPS 请求时，将发生以下操作：

1. DNS 将请求路由到适用边缘站点中您的分配的 IP 地址。
2. 如果客户端请求在 `ClientHello` 消息中提供 SNI 扩展，CloudFront 会搜索与该 SNI 关联的分配。
   - 如果存在匹配项，CloudFront 会使用 SSL/TLS 证书响应该请求。
   - 如果没有匹配项，CloudFront 会改用 IP 地址来识别您的分配，并确定将哪个 SSL/TLS 证书返回给查看器。
3. 查看器和 CloudFront 使用您的 SSL/TLS 证书执行 SSL/TLS 协商。
4. CloudFront 将请求的内容返回到查看器。

此方法适用于每个 HTTPS 请求，无论用户使用的是浏览器还是其他查看器。

请求使用三个或更多专用 IP SSL/TLS 证书的权限

如果您需要权限将三个或更多 SSL/TLS 专用 IP 证书与 CloudFront 永久关联，请执行以下步骤。有关 HTTPS 请求的更多信息，请参阅选择 CloudFront 如何处理 HTTPS 请求。

> **Note**
>
> 此过程在您的多个 CloudFront 分配之间使用三个或三个以上的专用 IP 证书。默认值是 2。请记住，您不能将多个 SSL 证书绑定到一个分配。
> 您一次只能将一个 SSL/TLS 证书与一个 CloudFront 分配关联。此数字是在您的所有 CloudFront 分配中可以使用的专用 IP SSL 证书的总数。

请求权限以在 CloudFront 分配中使用三个或更多证书

1. 转到支持中心并创建案例。
2. 注明您需要多少个证书的使用授权，并在请求中描述具体情况。我们将尽快更新您的账户。
3. 继续执行下一个过程。

在 CloudFront 中使用 SSL/TLS 证书的要求

本主题中介绍了 SSL/TLS 证书的要求。除非另有说明，否则这些要求均适用于以下两项：

- 在查看器和 CloudFront 之间使用 HTTPS 的证书
- 在 CloudFront 和您的源之间使用 HTTPS 的证书

主题

证书颁发者

建议您使用 AWS Certificate Manager(ACM) 颁发的证书。有关从 ACM 获取证书的信息，请参阅 AWS Certificate Manager 用户指南。要在 CloudFront 中使用 ACM 证书，请确保您在美国东部（弗吉尼亚州北部）区域 (us-east-1) 中请求（或导入）该证书。

CloudFront 与 Mozilla 支持相同的证书颁发机构 (CA)，因此如果您不使用 ACM，请使用 Mozilla 包含的 CA 证书列表中的 CA 颁发的证书。有关获取和安装证书的更多信息，请参考 HTTP 服务器软件文档和 CA 文档。

用于 AWS Certificate Manager 的 AWS 区域

要在 AWS Certificate Manager（ACM）中使用证书以要求在查看器和 CloudFront 之间使用 HTTPS，请确保您在美国东部（弗吉尼亚州北部）区域（us-east-1）中请求（或导入）该证书。

如果您需要在 CloudFront 和您的源之间使用 HTTPS，并且正在 Elastic Load Balancing 中使用负载均衡器作为源，则您可以在任何 AWS 区域中请求或导入证书。

证书格式

证书必须采用 X.509 PEM 格式。这是您使用 AWS Certificate Manager 时的原定设置格式。

中间证书

如果您使用的是第三方证书颁发机构 (CA)，请在 .pem 文件的证书链中列出所有中间证书，从为您的域签署证书的 CA 所颁发的证书开始。通常，您将在以适当的链顺序列出中间证书和根证书的 CA 网站上找到文件。

> **Important**
>
> 请不要包括以下内容：根证书，未在信任路径中的中间证书，或者 CA 的公有密钥证书。

示例如下：

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
```

密钥类型

CloudFront 支持 RSA 和 ECDSA 公有/私有密钥对。

CloudFront 支持使用 RSA 和 ECDSA 证书连接到查看器和源的 HTTPS 连接。借助 AWS Certificate Manager（ACM），您可以请求和导入 RSA 或 ECDSA 证书，然后将它们与您的 CloudFront 分配相关联。

有关 CloudFront 支持的您可以在 HTTPS 连接中协商的 RSA 和 ECDSA 密码的列表，请参阅the section called “查看器和 CloudFront 之间支持的协议和密码”和the section called “CloudFront 与源之间受支持的协议和密码”。

私有密钥

如果您使用的是来自第三方证书颁发机构 (CA) 的证书，请注意以下几点：

- 私有密钥必须匹配证书中的公有密钥。

- 私有密钥必须采用 PEM 格式。
- 私有密钥无法使用密码加密。

如果 AWS Certificate Manager (ACM) 提供了证书，则 ACM 不会发布该私有密钥。私有密钥存储在 ACM 中，供与 ACM 集成的 AWS 服务使用。

权限

您必须具备使用和导入 SSL/TLS 证书的权限。如果您使用 AWS Certificate Manager (ACM)，建议您使用 AWS Identity and Access Management 权限以限制访问证书。有关更多信息，请参阅《AWS Certificate Manager 用户指南》中的身份和访问权限管理。

证书密钥大小

CloudFront 支持的证书密钥大小取决于密钥和证书类型。

对于 RSA 证书：

CloudFront 支持 1024 位、2048 位、3072 位和 4096 位 RSA 密钥。用于 CloudFront 的 RSA 证书的最大密钥长度为 4096 位。

请注意，ACM 颁发密钥最多为 2048 位的 RSA 证书。要使用 3072 位或 4096 位 RSA 证书，您需要从外部获取该证书并将其导入 ACM，之后即可将其与 CloudFront 一起使用。

有关如何确定 RSA 密钥大小的信息，请参阅确定 SSL/TLS RSA 证书中公有密钥的大小。

对于 ECDSA 证书：

CloudFront 支持 256 位密钥。要在 ACM 中使用 ECDSA 证书以要求在查看器和 CloudFront 之间使用 HTTPS，请使用 prime256v1 椭圆曲线。

支持的证书类型

CloudFront 支持可信的证书颁发机构颁发的所有类型的证书。

证书到期日期和续订

如果您使用来自第三方证书颁发机构 (CA) 的证书，则您必须监控证书到期日期并在到期前续订导入到 AWS Certificate Manager (ACM) 或上载到 AWS Identity and Access Management 证书存储的证书。

如果您使用的是 ACM 提供的证书，则 ACM 会为您管理证书续订。有关更多信息，请参阅《AWS Certificate Manager 用户指南》中的托管续订。

CloudFront 分配和证书中的域名

当您使用自定义源时，源上 SSL/TLS 证书的公用名字段中包含域名，使用者备用名称字段中可能包含更多域名。（CloudFront 支持证书域名中的通配符。）

证书中的其中一个域名必须与您为“源域名”指定的域名匹配。如果没有任何域名匹配，CloudFront 会向查看器返回 HTTP 状态代码 502 (Bad Gateway)。

> **Important**
>
> 在将备用域名添加到分配时，CloudFront 会检查备用域名是否涵盖在您附加的证书中。证书必须在证书的使用者备用名称 (SAN) 字段中覆盖备用域名。这意味着，SAN 字段必须包含备用域名的完全匹配项，或者包含与要添加的备用域名级别相同的通配符。
> 有关更多信息，请参阅 使用备用域名的要求。

最低 SSL/TLS 协议版本

如果您使用的是专用 IP 地址，请通过选择安全策略来为查看器和 CloudFront 之间的连接选择最低 SSL/TLS 协议版本。

有关更多信息，请参阅 安全策略（最低 SSL/TLS 版本）主题中的 分配设置参考。

支持的 HTTP 版本

如果您将一个证书与多个 CloudFront 分配关联，则与证书关联的所有分配都必须针对支持的 HTTP 版本使用相同的选项。您在创建或更新 CloudFront 分配时指定此选项。

在 CloudFront 中使用 SSL/TLS 证书的配额（仅在查看器和 CloudFront 之间使用 HTTPS）

请注意在 CloudFront 中使用 SSL/TLS 证书的以下配额。这些配额仅适用于您使用 AWS Certificate Manager (ACM) 预置的 SSL/TLS 证书，或导入 ACM 或上传到 IAM 证书存储供查看器和 CloudFront 之间进行 HTTPS 通信的 SSL/TLS 证书。

有关更多信息，请参阅 增加 SSL/TLS 证书的配额。

每个 CloudFront 分配的最大证书数量

最多可以将一个 SSL/TLS 证书与每个 CloudFront 分配关联。

您可以导入 ACM 或上载到 IAM 证书存储的证书的最大数量

如果您从第三方 CA 获得了 SSL/TLS 证书，则必须在下列位置之一存储证书：

- AWS Certificate Manager – 有关 ACM 证书数量的当前配额，请参阅《AWS Certificate Manager 用户指南》中的配额。列出的限制是总数，包括您使用 ACM 预置的证书以及您导入 ACM 的证书。
- IAM 证书存储 – 有关您可以上载到 IAM 证书存储以供 AWS 账户使用的证书数的当前配额（以前称为限制），请参阅《IAM 用户指南》中的 IAM 和 STS 限制。您可以在 AWS Management Console中请求提高配额。

每个 AWS 账户的证书的最大数量（仅限专用 IP 地址）

如果要使用专用 IP 地址处理 HTTPS 请求，请注意以下几点：

- 默认情况下，CloudFront 向您授权以供您的AWS账户使用两个证书，一个用于日常使用，另一个用于您需要轮换证书以满足多个分配的情况。
- 如果您的 AWS 账户需要两个以上的自定义 SSL/TLS 证书，请转至支持中心并创建案例。注明您需要多少个证书的使用授权，并在请求中描述具体情况。我们将尽快更新您的账户。

对使用不同AWS账户创建的 CloudFront 分配使用同一个证书

如果您使用的是第三方 CA，并且希望针对不同 AWS 账户创建的多个 CloudFront 分配使用同一个证书，则必须将证书导入 ACM，或者为每个 AWS 账户将其上传至 IAM 证书存储一次。

如果您使用的是 ACM 提供的证书，则不能将 CloudFront 配置为使用其他 AWS 账户创建的证书。

为 CloudFront 和其他AWS服务使用同一个证书

如果您从信任的证书颁发机构 (如 Comodo、DigiCert 或 Symantec) 购买了证书，则可以对 CloudFront 和其他 AWS 服务使用同一个证书。如果要将证书导入到 ACM，您只需要导入一次即可供多个 AWS 服务使用。

如果您使用的是 ACM 提供的证书，这些证书会存储在 ACM 中。

为多个 CloudFront 分配使用同一个证书

对于任何或所有您正用来提供 HTTPS 请求的 CloudFront 分配，您都可以使用同一个证书。请注意以下几点：

- 您可以使用同一个证书来使用专用 IP 地址处理请求以及使用 SNI 处理请求。
- 只能将一个证书与每个分配关联。

- 每个分配必须包含一个或多个备用域名，这些域名也会出现在证书的公用名字段或主题备用名称字段中。
- 如果您正在使用专用 IP 地址提供 HTTPS 请求并且已使用同一个 AWS 账户创建您的所有分配，您可以为所有分配使用同一个证书，这样可以显著降低成本。CloudFront 对每个证书而不是每个分配收费。

  例如，假设您使用同一个 AWS 账户创建了三个分配，并且您对所有三个分配使用同一个证书。将仅针对您使用专用 IP 地址计算一次费用。

  但是，如果您正在使用专用 IP 地址处理 HTTPS 请求并且正在使用同一个证书在不同的 AWS 账户中创建 CloudFront 分配，则会向每个账户收取专用 IP 地址的使用费。例如，如果您使用三个不同的 AWS 账户创建三个分配，并且您为所有三个分配使用同一个证书，则向每个账户收取专用 IP 地址的完整使用费。

配置备用域名和 HTTPS

要在您的文件的 URL 中使用备用域名并在查看器和 CloudFront 之间使用 HTTPS，请执行适用步骤。

主题

获取 SSL/TLS 证书

如果您还没有 SSL/TLS 证书，请获取一个。有关更多信息，请参阅相应文档：

- 要使用 AWS Certificate Manager (ACM) 提供的证书，请参阅 AWS Certificate Manager 用户指南。然后跳至 更新 CloudFront 分配。

  > **Note**
  >
  > 建议您使用 ACM 在 AWS 托管资源上预置、管理和部署 SSL/TLS 证书。您必须在美国东部（弗吉尼亚州北部）区域申请 ACM 证书。

- 要从第三方证书颁发机构 (CA) 获取证书，请参阅该证书颁发机构提供的文档。当您拥有该证书时，请继续执行下一个步骤。

导入 SSL/TLS 证书

如果您从第三方 CA 得到证书，请将证书导入 ACM，或者上传到 IAM 证书存储：

ACM（推荐）

ACM 可让您从 ACM 控制台以及以编程方式导入第三方证书。有关将证书导入 ACM 的信息，请参阅《AWS Certificate Manager 用户指南》中的将证书导入 AWS Certificate Manager。您必须在美国东部（弗吉尼亚州北部）区域导入证书。

IAM 证书存储

（不推荐）请使用以下 AWS CLI 命令将您的第三方证书上载到 IAM 证书存储。

```
aws iam upload-server-certificate \
        --server-certificate-name CertificateName \
        --certificate-body file://public_key_certificate_file \
        --private-key file://privatekey.pem \
        --certificate-chain file://certificate_chain_file \
        --path /cloudfront/path/
```

请注意以下几点：

- AWS 账户 – 您必须使用创建 CloudFront 分配时使用的同一个 AWS 账户将证书上传到 IAM 证书存储。
- --path 参数 – 在将证书上传到 IAM 时，`--path` 参数（证书路径）的值必须以 `/cloudfront/` 开头，例如 `/cloudfront/production/` 或 `/cloudfront/test/`。该路径必须以 / 结尾。
- 现有的证书 – 您必须指定 `--server-certificate-name` 和 `--path` 参数的值，这些值不同于与现有证书关联的值。
- 使用 CloudFront 控制台 – 您在 AWS CLI 中为 `--server-certificate-name` 参数指定的值（例如 `myServerCertificate`）显示在 CloudFront 控制台的 SSL 证书列表中。
- 使用 CloudFront API – 记下 AWS CLI 返回的字母数字字符串，例如 `AS1A2M3P4L5E67SIIXR3J`。这是您将在 `IAMCertificateId` 元素中指定的值。您无需 IAM ARN（也由 CLI 返回）。

有关 AWS CLI 的更多信息，请参阅 AWS Command Line Interface 用户指南和 AWS CLI 命令参考。

更新 CloudFront 分配

要更新您的分配的设置，请执行以下步骤：

为备用域名配置您的 CloudFront 分配

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 为要更新的分配选择 ID。
3. 在 General 选项卡上，选择 Edit。
4. 更新以下值：

   备用域名（CNAME）

   选择添加项目，添加适用的备用域名。用逗号隔开多个域名，或在新行中键入每个域名。

   自定义 SSL 证书

   从下拉列表中选择证书。

   此处列出了多达 100 个证书。如果您有超过 100 个证书并且您没有看到要添加的证书，则可以在字段中键入一个证书 ARN 以选择它。

   如果您已将证书上传到 IAM 证书存储，但该证书没有被列出，并且您无法通过在字段中键入名称来选择它，请查看过程 导入 SSL/TLS 证书以确认您已正确上传证书。

   > **Important**
   >
   > 在将您的 SSL/TLS 证书与您的 CloudFront 分配关联之后，在您从所有分配中删除该证书并且已部署所有分配之前，请勿从 ACM 或 IAM 证书存储中删除该证书。

5. 选择保存更改。
6. 将 CloudFront 配置为要求查看器与 CloudFront 之间的通信为 HTTPS：

   a. 在行为选项卡中，选择要更新的缓存行为，然后选择编辑。

   b. 指定 Viewer Protocol Policy 的以下值之一：

将 HTTP 重定向到 HTTPS

查看器可使用两种协议，但 HTTP 请求将自动重定向到 HTTPS 请求。CloudFront 将返回 HTTP 状态代码 `301 (Moved Permanently)` 以及新的 HTTPS URL。然后，查看器会使用此 HTTPS URL 将请求重新提交到 CloudFront。

> **Important**
>
> CloudFront 不将 `DELETE`、`OPTIONS`、`PATCH`、`POST` 或 `PUT` 请求从 HTTP 重定向到 HTTPS。如果您将一个缓存行为配置为重定向到 HTTPS，CloudFront 会针对该缓存行为的 HTTP `DELETE`、`OPTIONS`、`PATCH`、`POST` 或 `PUT` 请求响应 HTTP 状态代码 `403 (Forbidden)`。

在查看器发出将重定向到 HTTPS 请求的 HTTP 请求时，会产生针对这两个请求的 CloudFront 费用。对于 HTTP 请求，仅对该请求和 CloudFront 返回到查看器的标头计费。对于 HTTPS 请求，对该请求、标头和由您的源返回的文件计费。

仅 HTTPS

查看器只有使用 HTTPS 才能访问您的内容。如果查看器发送 HTTP 请求而不是 HTTPS 请求，则 CloudFront 将返回 HTTP 状态代码 `403 (Forbidden)` 且不会返回此文件。

c. 选择是，编辑。

d. 针对要求在查看器和 CloudFront 之间使用 HTTPS 的其他每个缓存行为，重复步骤 a 到 c。

7. 请确认以下内容，然后在生产环境中使用更新后的配置：

- 每个缓存行为中的路径模式仅适用于您希望查看器使用 HTTPS 的请求。
- 缓存行为按您希望 CloudFront 评估它们的顺序列出。有关更多信息，请参阅 路径模式。
- 缓存行为将请求路由到正确的源。

确定 SSL/TLS RSA 证书中公有密钥的大小

如果您使用的是 CloudFront 备用域名和 HTTPS，则 SSL/TLS RSA 证书中公有密钥的最大大小为 4096 位。(这是密钥大小，不是指公有密钥中的字符数。) 在使用 AWS Certificate Manager 提供您的证书的情况下，虽然 ACM 支持更大的 RSA 密钥，但您无法在 CloudFront 中使用更大的密钥。

您可以通过运行以下 OpenSSL 命令来确定 RSA 公有密钥的大小：

```
openssl x509 -in path and filename of SSL/TLS certificate -text -noout
```

其中：

- `-in` 指定您的 SSL/TLS RSA 证书的路径和文件名。
- `-text` 使 OpenSSL 以位为单位显示 RSA 公有密钥的长度。
- `-noout` 阻止 OpenSSL 显示公有密钥。

输出示例：

```
Public-Key: (2048 bit)
```

增加 SSL/TLS 证书的配额

您可以导入到 AWS Certificate Manager（ACM）或上传至 AWS Identity and Access Management（IAM）的 SSL/TLS 证书的数量存在配额。在将 CloudFront 配置为使用专用 IP 地址处理 HTTPS 请求时，可以在一个AWS 账户中使用的 SSL/TLS 证书数也存在配额。但是，您可以请求提高配额。

主题

- 增加导入 ACM 的证书的配额
- 增加上传至 IAM 的证书的配额
- 增加用于专用 IP 地址的证书配额

增加导入 ACM 的证书的配额

有关可以导入到 ACM 的证书数量的配额，请参阅《AWS Certificate Manager 用户指南》中的配额。

要请求提高配额，请在支持中心控制台中创建案例。指定以下值：

- 接受提高服务限制的默认值。
- 对于限制类型，选择证书管理器。
- 对于区域，选择要导入证书的 AWS 区域。
- 对于限制，选择 ACM 证书数。

然后，填写表格的其余部分并提交。

增加上传至 IAM 的证书的配额

有关可上传到 IAM 的证书数量的配额（以前称为限制），请参阅《IAM 用户指南》中的 IAM 和 STS 限制。

要请求提高配额，请在支持中心控制台中创建案例。指定以下值：

- 接受提高服务限制的默认值。
- 对于限制类型，选择证书管理器。
- 对于区域，选择要导入证书的 AWS 区域。
- 对于限制，请选择服务器证书限制(IAM)。

然后，填写表格的其余部分并提交。

增加用于专用 IP 地址的证书配额

在使用专用 IP 地址处理 HTTPS 请求时，有关可用于每个AWS 账户的 SSL 证书的数量配额，请参阅 SSL 证书的配额。

要请求提高配额，请在支持中心控制台中创建案例。指定以下值：

- 接受提高服务限制的默认值。
- 对于限制类型，请选择 CloudFront 分配。
- 对于限制，选择每个账户的专用 IP SSL 证书限制。

然后，填写表格的其余部分并提交。

轮换 SSL/TLS 证书

如果您使用的是 AWS Certificate Manager (ACM) 提供的证书，则无需轮换 SSL/TLS 证书。ACM 为您管理证书的续订。有关更多信息，请参阅《AWS Certificate Manager用户指南》中的托管续订。

> **Note**
>
> 对于您从第三方证书颁发机构获取并导入 ACM 的证书，ACM 不会为其管理证书续订。

如果您使用的是第三方证书颁发机构，并且已将证书导入到 ACM（推荐）或上传到 IAM 证书存储中，则有时必须将一个证书更换为另一个证书。例如，您必须在即将到达证书的到期日期时替换证书。

> **Important**
>
> 如果您将 CloudFront 配置为使用专用 IP 地址提供 HTTPS 请求，就您在轮换证书时使用一个或多个其他证书，可能会产生额外的按比例分摊的费用。建议您及时更新您的分配，以最大程度地降低费用。

轮换 SSL/TLS 证书

要轮换证书，请执行以下步骤。在您轮换证书以及轮换过程完成时，查看器可以继续访问您的内容。

轮换 SSL/TLS 证书

1. 增加 SSL/TLS 证书的配额以确定您是否需要权限来使用更多的 SSL 证书。如果需要，可请求权限并等待授予权限，然后继续执行步骤 2。
2. 将新证书导入 ACM 或者上传到 IAM。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的导入 SSL/TLS 证书。
3. 请一次更新一项分配，使其使用新证书。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的列出、查看和更新 CloudFront 分配。
4. （可选）更新您的所有 CloudFront 分配之后，您可以从 ACM 或 IAM 中删除旧证书。

> **Important**
>
> 在您将 SSL/TLS 证书从所有分配中删除并且您所更新的分配的状态变为 `Deployed` 前，请勿删除该证书。

从自定义 SSL/TLS 证书恢复为默认 CloudFront 证书

如果您将 CloudFront 配置为在查看器和 CloudFront 之间使用 HTTPS，并将 CloudFront 配置为使用自定义 SSL/TLS 证书，则可以更改配置以使用默认的 CloudFront SSL/TLS 证书。此过程取决于您是否已使用分配来分发内容：

- 如果您尚未使用分配来分发内容，则只需更改配置。有关更多信息，请参阅 更新分配。

- 如果您已使用分配来分发内容，则必须创建新的 CloudFront 分配并更改文件的 URL，以便减少或消除内容不可用的时间。为此，请执行以下步骤。

恢复到默认 CloudFront 证书

以下步骤演示如何从自定义 SSL/TLS 证书恢复为默认 CloudFront 证书。

恢复到默认 CloudFront 证书

1. 采用所需的配置创建一项新的 CloudFront 分配。对于 SSL 证书，请选择默认 CloudFront 证书 (*.cloudfront.net)。

   有关更多信息，请参阅 创建分配。

2. 对于使用 CloudFront 分配的文件，请将您应用程序中的 URL 更新为使用 CloudFront 为新分配指定的域名。例如，将 `https://www.example.com/images/logo.png` 更改为 `https://d111111abcdef8.cloudfront.net/images/logo.png`。

3. 删除与自定义 SSL/TLS 证书关联的分配，或者更新分配以将 SSL 证书值更改为默认 CloudFront 证书 (*.cloudfront.net)。有关更多信息，请参阅 更新分配。

   > **Important**
   >
   > 在您完成此步骤之前，AWS 会继续向您收取使用自定义 SSL/TLS 证书的费用。

4. （可选）删除您的自定义 SSL/TLS 证书。

   a. 运行 AWS CLI 命令 `list-server-certificates` 以获取要删除的证书的证书 ID。有关更多信息，请参阅 AWS CLI命令参考中的 list-server-certificates。

   b. 运行 AWS CLI 命令 `delete-server-certificate` 以删除该证书。有关更多信息，请参阅《AWS CLI 命令参考》中的 delete-server-certificate。

将使用专用 IP 地址的自定义 SSL/TLS 证书切换到 SNI

如果您已将 CloudFront 配置为使用自定义 SSL/TLS 证书及专用 IP 地址，可以改为切换到使用自定义 SSL/TLS 证书及 SNI，并消除与专用 IP 地址关联的费用。以下步骤将说明如何操作。

> **Important**
>
> 对您的 CloudFront 配置进行此更新不会影响支持 SNI 的查看器。查看器既可以在更改前后访问您的内容，也可以在将更改传播到 CloudFront 边缘站点时访问您的内容。不支持 SNI 的查看器在更改后无法访问您的内容。有关更多信息，请参阅 选择 CloudFront 如何处理 HTTPS 请求。

从自定义证书切换到 SNI

以下步骤演示如何将使用专用 IP 地址的自定义 SSL/TLS 证书切换到 SNI。

使用专用 IP 地址从自定义 SSL/TLS 证书切换到 SNI

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择要查看或更新的分配的 ID。
3. 选择分配设置。
4. 在 General 选项卡上，选择 Edit。
5. 将自定义 SSL 客户端支持设置更改为仅限支持服务器名称指示 (SNI) 的客户端。
6. 选择是，编辑。

使用签名 URL 和签名 Cookie 提供私有内容

许多通过互联网分发内容的公司都希望限制对文档、业务数据、流媒体或面向选定用户（例如付费用户）的内容的访问。要使用 CloudFront 安全地提供这种私有内容，可执行以下操作：

- 要求用户使用特殊的 CloudFront 签名 URL 或签名 Cookie 访问私有内容。
- 要求您的用户使用 CloudFront URL 访问内容，而不是直接用源服务器（例如 Amazon S3 或私有 HTTP 服务器）上的 URL 访问内容。CloudFront URL 不是必需的，但建议使用，以防止用户绕过在已签名的 URL 或已签名的 Cookie 中指定的限制。

有关更多信息，请参阅 限制对文件的访问。

如何提供私有内容

要配置 CloudFront 以提供私有内容，请执行以下任务。

1. （可选但建议使用）要求用户仅通过 CloudFront 访问您的内容。使用的方法取决于您使用的是 Amazon S3 还是自定义源：

   - Amazon S3 – 请参阅the section called “限制对 Amazon Simple Storage Service 源的访问”。
   - 自定义源 – 请参阅在自定义源上限制对文件的访问。

   自定义源包括 Amazon EC2、配置为网站端点的 Amazon S3 存储桶、Elastic Load Balancing 以及您自己的 HTTP Web 服务器。
2. 指定您希望用于创建签名 URL 或签名 Cookie 的可信密钥组 或可信签署人。建议您使用可信密钥组。有关更多信息，请参阅 指定可以创建签名 URL 和签名 Cookie 的签署人。
3. 将应用程序编写为响应来自授权用户的请求，这些用户使用签名 URL 或使用设置签名 Cookie 的 `Set-Cookie` 标头。请按照以下主题之一中的步骤操作：

   - 使用签名 URL
   - 使用签名 Cookie

   如果您不确定要使用哪种方法，请参阅决定使用签名 URL 还是签名 Cookie。

主题

限制对文件的访问

您可通过两种方式控制用户对私有内容的访问：

- 限制对 CloudFront 缓存中的文件的访问。
- 通过执行下列操作之一，限制对您源中文件的访问：
  - 为 Amazon S3 存储桶设置源访问控制 (OAI)。

- 为私有 HTTP 服务器（自定义源）配置自定义标头。

限制对 CloudFront 缓存中的文件的访问

您可以将 CloudFront 配置为要求用户使用签名 URL 或签名 Cookie 访问您的文件。然后开发应用程序，以创建签名 URL 并将其分发给经身份验证的用户，或者为经身份验证的用户发送用于设置签名 Cookie 的 `Set-Cookie` 标头。（要为一些用户提供长期访问少量文件的权限，还可以手动创建签名 URL。）

创建签名 URL 或签名 Cookie 以控制对您的文件的访问时，可以指定以下限制：

- 结束日期和时间，在此之后，URL 不再有效。
- （可选）URL 生效的日期和时间。
- （可选）可用于访问您的内容的 IP 地址或计算机的地址范围。

签名 URL 或签名 Cookie 的其中一部分使用公有/私有密钥对中的私有密钥进行哈希处理和签名。当某人使用签名 URL 或签名 Cookie 访问文件时，CloudFront 将比较 URL 或 Cookie 的已签名部分和未签名部分。如果它们不匹配，则 CloudFront 将不提供该文件。

您必须使用 RSA-SHA1 对 URL 或 Cookie 进行签名。CloudFront 不接受其他算法。

限制对 Amazon S3 存储桶中文件的访问

您可以选择保护 Amazon S3 存储桶中的内容，以便用户可以通过指定的 CloudFront 分配访问内容，但不能使用 Amazon S3 URL 直接访问内容。这可防止其他人绕过 CloudFront 并使用 Amazon S3 URL 访问您希望限制访问的内容。虽然此步骤未要求使用签名 URL，但我们建议使用。

如需要求用户通过 CloudFront URL 访问内容，请执行以下任务：

- 为 CloudFront 源访问控制授予读取 S3 存储桶中的文件的权限。
- 创建源访问控制，并将其与您的 CloudFront 分配相关联。
- 删除其他任何人使用 Amazon S3 URL 读取这些文件的权限。

有关更多信息，请参阅 the section called “限制对 Amazon Simple Storage Service 源的访问”。

在自定义源上限制对文件的访问

如果您使用自定义源，则可以选择设置自定义标头来限制访问。要使 CloudFront 从自定义源获取文件，CloudFront 必须使用标准 HTTP（或 HTTPS）请求访问这些文件。但是，使用自定义标头，您可以进一步限制对内容的访问，使得用户只能通过 CloudFront 进行访问而无法直接访问。虽然此步骤未要求使用签名 URL，但我们建议使用。

如需要求用户通过 CloudFront 访问内容，请在 CloudFront 分配中更改以下设置：

源自定义标头

将 CloudFront 配置为将自定义标头转发到源。请参阅 配置 CloudFront 以便向源请求添加自定义标头。

查看器协议策略

将分配配置为要求查看器使用 HTTPS 访问 CloudFront。请参阅 查看器协议策略。

源协议策略

将分配配置为要求 CloudFront 与查看器使用相同协议来向源转发请求。请参阅 协议（仅自定义源）。

进行了这些更改之后，在自定义源上更新应用程序，以仅接受满足如下条件的请求：其中包含您已将 CloudFront 配置为发送的自定义标头。

查看器协议策略和源协议策略的组合可确保在传输过程中对自定义标头进行加密。但是，建议您定期执行以下任务来轮换 CloudFront 转发到源的自定义标头：

1. 更新 CloudFront 分配，开始将新标头转发至自定义源。
2. 更新应用程序以接受新标头，从而确认请求来自 CloudFront。
3. 当请求不再包含您要替换的标头时，请更新应用程序以便不再接受旧标头，从而确认请求来自 CloudFront。

指定可以创建签名 URL 和签名 Cookie 的签署人

主题

- 在可信密钥组（推荐）和AWS 账户之间进行选择
- 为签署人创建密钥对

- 重新设置私有密钥的格式（仅限 .NET 和 Java）
- 将签署人添加到分配
- 轮换密钥对

要创建签名 URL 或签名 Cookie，您需要一个签署人。签署人可以是您在 CloudFront 中创建的受信密钥组，也可以是包含 CloudFront 密钥对的 AWS 账户。建议您使用带有签名 URL 和签名 Cookie 的可信密钥组。有关更多信息，请参阅 在可信密钥组（推荐）和AWS 账户之间进行选择。

签署人有两个目的：

- 只要将可信签署人添加到分配中，CloudFront 就会开始要求查看器使用签名 URL 或签名 Cookie 访问文件。
- 创建签名 URL 或签名 Cookie 时，使用来自签署人的密钥对中的私有密钥来签署 URL 或 Cookie 的一部分。当有人请求受限文件时，CloudFront 会将 URL 或 Cookie 中的签名与未签名 URL 或 Cookie 进行比较，以确认其未被篡改。CloudFront 还会验证 URL 或 Cookie 是否有效，即，未超过过期日期和时间。

指定签署人时，还可以通过将签署人添加到缓存行为，间接指定需要签名 URL 或签名 Cookie 的文件。如果分配只有一个缓存行为，则查看器必须使用签名 URL 或签名 Cookie 访问分配中的任何文件。如果创建了多个缓存行为，并将签署人添加到某些缓存行为而没有添加到其他缓存行为，则可要求查看器使用签名 URL 或签名 Cookie 访问某些文件而不是其他文件。

要指定允许创建签名 URL 或签名 Cookie 的签署人（私有密钥）并将签署人添加到 CloudFront 分配中，请执行以下任务：

1. 决定是使用可信密钥组还是AWS 账户作为签署人。我们建议使用可信密钥组。有关更多信息，请参阅 在可信密钥组（推荐）和AWS 账户之间进行选择。
2. 对于您在步骤 1 中选择的签署人，创建一个公有/私有密钥对。有关更多信息，请参阅 为签署人创建密钥对。
3. 如果使用 .NET 或 Java 创建签名 URL 或签名 Cookie，请重新设置私有密钥的格式。有关更多信息，请参阅 重新设置私有密钥的格式（仅限 .NET 和 Java）。
4. 在要为其创建签名 URL 或签名 Cookie 的分配中，指定签署人。有关更多信息，请参阅 将签署人添加到分配。

在可信密钥组（推荐）和AWS 账户之间进行选择

要使用签名 URL 或签名 Cookie，您需要一个签署人。签署人可以是您在 CloudFront 中创建的可信密钥组，也可以是包含 CloudFront 密钥对的AWS 账户。建议您使用可信密钥组，原因如下：

- 对于 CloudFront 密钥组，您无需使用 AWS 账户 root 用户来管理 CloudFront 签名 URL 和签名 Cookie 的公有密钥。AWS最佳实践建议您在不必使用 root 用户时就不要使用。
- 借助 CloudFront 密钥组，您可以使用 CloudFront API 管理公有密钥、密钥组和可信签署人。您可以使用 API 自动执行密钥创建和密钥轮换。当您使用 AWS root 用户时，您必须使用 AWS Management Console管理 CloudFront 密钥对，因此您无法自动执行此过程。
- 由于您可以使用 CloudFront API 管理密钥组，因此还可以使用 AWS Identity and Access Management (IAM) 权限策略来限制允许不同用户执行的操作。例如，您可以允许用户上传公有密钥，但不能删除它们。或者，您可以允许用户删除公有密钥，但只有在满足某些条件时才能删除公有密钥，例如使用多重验证、从特定网络发送请求或在特定日期和时间范围内发送请求。
- 通过 CloudFront 密钥组，您可以将更多的公有密钥与您的 CloudFront 分配关联，从而在如何使用和管理公有密钥方面提供更大的灵活性。默认情况下，您最多可以将四个密钥组与单一分配关联，并且一个密钥组中最多可以有五个公有密钥。

  当您使用 AWS 账户根用户管理 CloudFront 密钥对时，每个 AWS 账户最多只能拥有两个有效的 CloudFront 密钥对。

为签署人创建密钥对

您用于创建 CloudFront 签名 URL 或签名 Cookie 的每个签署人都必须具有公有密钥/私有密钥对。签署人使用其私有密钥对 URL 或 Cookie 进行签名，而 CloudFront 使用公有密钥验证签名。

创建密钥对的方式取决于您是使用可信密钥组作为签署人（推荐），还是使用 CloudFront 密钥对。有关更多信息，请参阅以下部分。您创建的密钥对必须满足以下要求：

- 它必须是 SSH-2 RSA 密钥对。
- 它必须采用 base64 编码的 PEM 格式。
- 它必须是 2048 位密钥对。

为了帮助保护您的应用程序，建议您定期轮换密钥对。有关更多信息，请参阅 轮换密钥对。

为可信密钥组创建密钥对（推荐）

要为可信密钥组创建密钥对，请执行以下步骤：

1. 创建公有/私有密钥对。
2. 将公有密钥上传到 CloudFront。
3. 将公有密钥添加到 CloudFront 密钥组。

有关更多信息，请参阅以下流程。

创建密钥对

> **Note**
>
> 以下步骤使用 OpenSSL 作为一种密钥对创建方法的示例。还有许多其他方法可以创建 RSA 密钥对。

1. 以下示例命令使用 OpenSSL 生成长度为 2048 位的 RSA 密钥对，并将其保存到名为 `private_key.pem` 的文件中。

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

2. 生成的文件同时包含公有密钥和私有密钥。以下示例命令从名为 `private_key.pem` 的文件中提取公有密钥。

   ```
   openssl rsa -pubout -in private_key.pem -out public_key.pem
   ```

   您稍后在以下过程中上传公有密钥（在 `public_key.pem` 文件中）。

将公有密钥上传到 CloudFront

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航菜单中，选择公有密钥。
3. 选择创建公有密钥。
4. 在创建公有密钥窗口中，执行以下操作：

   a. 对于密钥名称，键入一个名称以标识公有密钥。

b. 对于密钥值，粘贴公有密钥。如果按照上述过程中的步骤操作，则公有密钥位于名为 `public_key.pem` 的文件中。要复制和粘贴公有密钥的内容，您可以：

- 在 macOS 或 Linux 命令行上使用 cat 命令，如下所示：

```
cat public_key.pem
```

复制该命令的输出，然后将其粘贴到密钥值字段中。

- 使用记事本（在 Windows 上）或 TextEdit（在 macOS 上）等明文编辑器打开 `public_key.pem` 文件。复制文件的内容，然后将其粘贴到密钥值字段中。

c. （可选）对于注释，请添加注释以描述公有密钥。

完成后，选择添加。

5. 记录公有密钥 ID。稍后在创建签名 URL 或签名 Cookie 时使用它作为 `Key-Pair-Id` 字段的值。

将公有密钥添加到密钥组

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 在导航菜单中，选择密钥组。
3. 选择添加密钥组。
4. 在创建密钥组页上，执行以下操作：

a. 对于密钥组名称，键入一个名称以标识密钥组。

b. （可选）对于注释，键入注释以描述密钥组。

c. 对于公有密钥，选择要添加到密钥组的公有密钥，然后选择添加。对要添加到密钥组的每个公有密钥重复此步骤。

5. 选择创建密钥组。
6. 记录密钥组名称。稍后可以使用它将密钥组与 CloudFront 分配中的缓存行为相关联。（在 CloudFront API 中，您使用密钥组 ID 将密钥组与缓存行为相关联。）

创建 CloudFront 密钥对（不建议这样做，需要AWS 账户根用户权限）

> **Important**
>
> 建议您为可信密钥组创建公有密钥，而不是执行以下步骤。有关为签名 URL 和签名 Cookie 创建公有密钥的推荐方法，请参阅为可信密钥组创建密钥对（推荐）。

您可以通过以下方式创建 CloudFront 密钥对：

- 在 AWS Management Console中创建密钥对并下载私有密钥。请参见以下过程。
- 通过使用诸如 OpenSSL 等应用程序创建 RSA 密钥对，然后将公有密钥上传到 AWS Management Console。有关创建 RSA 密钥对的更多信息，请参阅为可信密钥组创建密钥对（推荐）。

要在 AWS Management Console中创建 CloudFront 密钥对

1. 使用 AWS 账户 root 用户登录 AWS Management Console。

   > **Important**
   >
   > IAM 用户无法创建 CloudFront 密钥对。必须使用 root 用户凭证登录才能创建密钥对。

2. 选择您的账户名称，然后选择我的安全凭证。
3. 选择 CloudFront 密钥对。
4. 确认没有或仅拥有一个有效的密钥对。如果已有两个有效的密钥对，则无法创建密钥对。
5. 选择创建新的密钥对。

   > **Note**
   >
   > 您还可以选择创建自己的密钥对并上传公钥。CloudFront 密钥对支持 1024、2048 或 4096 位密钥。

6. 在创建密钥对对话框中，选择下载私有密钥文件，然后将该文件保存在您的计算机上。

   > **Important**
   >
   > 将 CloudFront 密钥对的私有密钥保存在安全的位置，并设置对文件的权限，以便只有所需的管理员可读取它。如果某人获取您的私有密钥，则他们可以生成有效的签名 URL 和签

名 Cookie，并下载您的内容。您不能再次获得私有密钥，因此，如果您丢失或删除了它，则必须创建新的 CloudFront 密钥对。

7. 记录密钥对的密钥对 ID。(在 AWS Management Console 中，这称为访问密钥 ID。) 创建签名 URL 或签名 Cookie 时会用到它。

重新设置私有密钥的格式（仅限 .NET 和 Java）

如果使用 .NET 或 Java 创建签名 URL 或签名 Cookie，则不能以默认 PEM 格式使用密钥对中的私有密钥来创建签名。而是执行以下操作：

- .NET 框架 – 将私有密钥转换成 .NET 框架使用的 XML 格式。可使用多种工具。
- Java – 将私有密钥转换成 DER 格式。执行此操作的一种方法是使用以下 OpenSSL 命令。在以下命令中，`private_key.pem` 是包含 PEM 格式的私有密钥的文件的名称，`private_key.der` 是运行该命令后包含 DER 格式的私有密钥的文件的名称。

```
openssl pkcs8 -topk8 -nocrypt -in private_key.pem -inform PEM -out private_key.der -
outform DER
```

要确保编码器正常工作，将 Bouncy Castle Java 加密术 API 的 JAR 添加到您的项目中，然后添加 Bouncy Castle 提供商。

将签署人添加到分配

签署人是可以为分配创建签名 URL 和签名 Cookie 的可信密钥组（推荐）或 CloudFront 密钥对。要在 CloudFront 分配中使用签名 URL 或签名 Cookie，您必须指定签署人。

签署人与缓存行为相关联。这样，您就可以要求对同一分配中的某些文件使用签名 URL 或签名 Cookie，对另一些文件则不使用。仅对于与相应缓存行为关联的文件，分配才需要签名 URL 或 Cookie。

同样，签署人只能为与相应缓存行为相关联的文件签署 URL 或 Cookie。例如，如果您有两个签署人，他们分别针对两个不同的缓存行为，那么这两个签署人均不能为与另一个缓存行为相关联的文件创建签名 URL 或 Cookie。

Important

在将签署人添加到分配之前，请执行以下操作：

- 仔细定义缓存行为中的路径模式和缓存行为的顺序，以便不会授予用户意外访问您的内容的权限或阻止他们访问您希望每个人都可用的内容。

  例如，假设请求匹配两个缓存行为的路径模式。第一个缓存行为不要求签名 URL 或签名 Cookie，而第二个缓存行为有此要求。这种情况下，用户不需使用签名 URL 或签名 Cookie 即可访问文件，因为 CloudFront 处理的缓存行为与第一个相符的条件相关联。

  更多有关路径模式的信息，请参阅 路径模式。
- 对于您已经用于分发内容的分配，请确保您已准备好开始生成签名 URL 和签名 Cookie，然后再添加签署人。添加签署人时，CloudFront 拒绝不包含有效签名 URL 或签名 Cookie 的请求。

您可以使用 CloudFront 控制台或 CloudFront API 将签名者添加到您的分配中。

Console

以下步骤显示如何将可信密钥组添加为签署人。您也可以将AWS 账户添加为可信签署人，但不建议这样做。

使用控制台将签署人添加到分配

1. 记录要用作可信签署人的密钥组的密钥组 ID。有关更多信息，请参阅 为可信密钥组创建密钥对（推荐）。
2. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
3. 选择要使用签名 URL 或签名 Cookie 保护其文件的分配。

   Note

   要将签署人添加到新分配，请在创建分配时指定在步骤 6 中描述的相同设置。
4. 选择 Behaviors 选项卡。
5. 选择其路径模式与要使用签名 URL 或签名 Cookie 保护的文件相匹配的缓存行为，然后选择编辑。
6. 在编辑行为页上，执行以下操作：

a. 对于限制查看器访问(使用签名 URL 或签名 Cookie)，选择是。

b. 对于可信密钥组或可信签署人，选择可信密钥组。

c. 对于可信密钥组，选择要添加的密钥组，然后选择添加。如果要添加多个密钥组，请重复此操作。

7. 选择是，编辑以更新缓存行为。

API

您可以使用 CloudFront API 将可信密钥组添加为签署人。您可以将签署人添加到现有分配或新分配。在这两种情况下，在 `TrustedKeyGroups` 元素中指定适用的值。

您也可以将AWS 账户添加为可信签署人，但不建议这样做。

请参阅《Amazon CloudFront API 参考》中的以下主题：

- 更新现有的分配 – UpdateDistribution
- 创建新的分配 – CreateDistribution

轮换密钥对

建议您定期轮换（更改）签名 URL 和签名 Cookie 的密钥对。要轮换用于创建签名 URL 或签名 Cookie 的密钥对，而不使尚未过期的 URL 或 Cookie 失效，请执行以下任务：

1. 创建新的密钥对，然后将公有密钥添加到密钥组。有关更多信息，请参阅 为可信密钥组创建密钥对（推荐）。
2. 如果您在上一步骤中创建了新密钥组，请将密钥组作为签署人添加到分配中。

> **Important**
>
> 不要从密钥组中删除任何现有公有密钥，也不要从分配中删除任何密钥组。只添加新的密钥组。

3. 更新您的应用程序，以使用新密钥对中的私有密钥创建签名。确认使用新私有密钥签署的签名 URL 或 Cookie 有效。
4. 等待使用之前的私有密钥对签名的 URL 或 Cookie 中的过期日期已过。然后，从密钥组中删除旧公有密钥。如果您在步骤 2 中创建了新的密钥组，请从分配中删除旧密钥组。

决定使用签名 URL 还是签名 Cookie

CloudFront 签名 URL 和签名 Cookie 提供相同的基本功能：它们允许您控制哪些用户可访问您的内容。如果您希望通过 CloudFront 提供私有内容，但尚未决定是使用签名 URL 还是签名 Cookie，请考虑以下因素。

在以下情况下使用签名 URL：

- 您希望限制对单个文件的访问，例如应用程序的安装程序下载。
- 用户使用不支持 Cookie 的客户端 (例如，自定义 HTTP 客户端)。

在以下情况下使用签名 Cookie：

- 您希望提供对多个限制文件的访问，例如，HLS 格式视频的所有文件或者网站订户区域中的所有文件。
- 您不希望更改当前 URL。

如果当前未使用签名 URL，并且未签名 URL 包含以下任一查询字符串参数，则不能使用签名 URL 或签名 Cookie：

- `Expires`
- `Policy`
- `Signature`
- `Key-Pair-Id`

CloudFront 假定包含任何这些查询字符串参数的 URL 是签名 URL，因此不会再查看签名 Cookie。

同时使用签名 URL 和签名 Cookie

签名 URL 优先于签名 Cookie。如果同时使用签名 URL 和签名 Cookie 来控制对相同文件的访问，并且查看器使用签名 URL 来请求文件，则 CloudFront 将仅基于签名 URL 确定是否向查看器返回文件。

使用签名 URL

签名 URL 包括额外的信息，例如，过期日期和时间，为您提供内容访问方面的更多控制权。该额外信息出现在策略声明中，且是基于标准策略或自定义策略。标准策略和自定义策略之间的差别将在接下来的两节中予以说明。

> **Note**
>
> 针对相同分配，您可以使用标准策略创建一些签名 URL 以及使用自定义策略创建一些签名 URL。

主题

决定为签名 URL 使用标准策略还是自定义策略

创建签名 URL 时，需要编写 JSON 格式的策略声明，以指定对签名 URL 的限制，例如，URL 的有效期。可以使用标准策略或自定义策略。以下是标准策略和自定义策略的比较：

说明	标准策略	自定义策略
可对多个文件重复使用策略声明。要重复使用策略声明，您必须在 `Resource` 对象中使用通配符。有关更多信息，请参阅在使用自定义策略的签名 URL 的策略声明中指定的值。	否	是
可指定用户开始访问内容的日期和时间。	否	是（可选）
可指定用户无法再访问内容的日期和时间。	是	是
可指定能够访问内容的用户的 IP 地址或 IP 地址范围。	否	是（可选）
签名 URL 包括策略的 Base64 编码版本，这会导致更长的 URL。	否	是

有关使用标准策略创建签名 URL 的信息，请参阅使用标准策略创建签名 URL。

有关使用自定义策略创建签名 URL 的信息，请参阅使用自定义策略创建签名 URL。

签名 URL 的工作方式

以下概述了您如何为签名 URL 配置 CloudFront 和 Amazon S3，以及在用户使用签名 URL 请求文件时 CloudFront 如何响应。

1. 在 CloudFront 分配中，指定一个或多个可信密钥组，这些密钥组包含 CloudFront 可用来验证 URL 签名的公有密钥。您可以使用相应的私有密钥对 URL 进行签名。

   有关更多信息，请参阅 指定可以创建签名 URL 和签名 Cookie 的签署人。
2. 开发应用程序，以确定用户是否应拥有对您的内容的访问权，以及为您希望限制访问的文件或应用程序的某些部分创建签名 URL。有关更多信息，请参阅以下主题：

   - 使用标准策略创建签名 URL
   - 使用自定义策略创建签名 URL
3. 用户请求您要求对其使用签名 URL 的文件。
4. 应用程序验证用户是否有权访问文件：他们已登录、已付费访问内容或已满足一些其他访问要求。
5. 您的应用程序创建并返回签名 URL 给用户。
6. 签名 URL 允许用户下载或流式传输内容。

   此步骤是自动的；用户通常不必做任何额外的事情以访问内容。例如，如果用户是在 Web 浏览器中访问您的内容，那么您的应用程序会将签名 URL 发回浏览器。浏览器立即在无用户干预的情况下使用签名 URL 访问 CloudFront 边缘缓存中的文件。
7. CloudFront 使用公有密钥验证签名并确认该 URL 未被篡改。如果签名无效，则请求将被拒绝。

   如果签名有效，CloudFront 将查看 URL 中的策略声明（如果使用标准策略，则构造一个），以确认该请求仍然有效。例如，如果您为该 URL 指定了开始和结束日期及时间，CloudFront 会确认用户是否是在您希望允许访问的时间段尝试访问您的内容。

   如果请求满足策略声明中的要求，CloudFront 将执行标准操作：确定文件是否已位于边缘缓存中，必要时将请求转发到源，然后向用户返回文件。

> **Note**
>
> 如果未签名的 URL 包含查询字符串参数，请确保将它们包含在 URL 中您签名的部分内。如果在签署后将查询字符串添加到签名 URL 中，该 URL 将返回 HTTP 403 状态。

决定签名 URL 的有效时间长度

您可使用只在很短时间内有效（可能只有几分钟）的签名 URL 来分配私有内容。有效时间如此短的签名 URL 适用于出于特定的目的即时向用户分发内容，如按需向客户分发租赁的电影或下载的音乐。如果您的签名 URL 有效期较短，您将可能希望使用您开发的应用程序自动生成它们。当用户开始下载文件或开始播放媒体文件时，CloudFront 将比较 URL 中的过期时间和当前时间，以确定 URL 是否仍然有效。

您也可使用有效时间较长 (可能数年) 的签名 URL 来分配私有内容。有效时间较长的签名 URL 适用于向已知用户分发私有内容，如向投资者分发业务计划或向员工分发培训材料等。您可以开发一个应用程序来为您生成这些长期签名 URL。

CloudFront 何时检查签名 URL 中的过期日期和时间

在发出 HTTP 请求时，CloudFront 检查签名 URL 中的过期日期和时间。如果客户端刚好在过期时间之前开始下载大型文件，即使在下载过程中到了过期时间，该下载也应该完成。如果 TCP 连接断开，并且客户端试图在过期时间到期后重新开始下载，则下载将会失败。

如果客户端使用 Range GET 来获取较小的文件，在过期时间到期后发生的任何 GET 请求将会失败。有关 Range GET 的更多信息，请参阅 CloudFront 如何处理对象的部分请求 (Range GET)。。

代码示例和第三方工具

有关创建签名 URL 的散列和签署部分的代码示例，请参阅以下主题：

- 使用 Perl 创建 URL 签名
- 使用 PHP 创建 URL 签名
- 使用 C# 和 .NET Framework 创建 URL 签名
- 使用 Java 创建 URL 签名

使用标准策略创建签名 URL

要使用标准策略创建签名 URL，请完成以下步骤。

要使用标准策略创建签名 URL

1. 如果您使用 .NET 和 Java 创建签名 URL，而且，如果您尚未将密钥对私有密钥的格式从默认 .pem 格式重新设置为与 .NET 和 Java 兼容的格式，那么现在就开始设置吧。有关更多信息，请参阅 重新设置私有密钥的格式（仅限 .NET 和 Java）。
2. 按照列出的顺序连接以下值，复制本示例签名 URL 中显示的格式：

```
https://d111111abcdef8.cloudfront.net/
image.jpg?color=red&size=medium&Expires=1357034400&Signature=nitfHRCrtziwO2HwPfWw~yYDhUF5Ew
j19DzZrvDh6hQ73lDx~-ar3UocvvRQVw6EkC~GdpGQyyOSKQim-
TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmatEXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6&Key-
Pair-Id=K2JCJMDEHXQW5F
```

删除所有空格（包括制表符和换行符）。您可能需要在应用程序代码的字符串中包括换码符。所有值的类型均为 String。

1. ***##### URL***

   基本 URL 是您将用于访问文件的 CloudFront URL，如果您不使用签名 URL，包括您自己的查询字符串参数（如果有）。在上一示例中，基本 URL 为 https://d111111abcdef8.cloudfront.net/image.jpg。有关适用于分配的 URL 格式的更多信息，请参阅在 CloudFront 中自定义文件的 URL 格式。

   - 以下 CloudFront URL 适用于分配中的图像文件（使用 CloudFront 域名）。请注意，image.jpg 是在 images 目录中。URL 中文件的路径必须与您的 HTTP 服务器或 Amazon S3 存储桶中文件的路径匹配。

     ```
     https://d111111abcdef8.cloudfront.net/images/image.jpg
     ```

   - 以下 CloudFront URL 包含查询字符串：

     ```
     https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large
     ```

   - 以下 CloudFront URL 适用于分配中的映像文件。均使用备用域名。第二个包括查询字符串：

     ```
     https://www.example.com/images/image.jpg
     ```

     ```
     https://www.example.com/images/image.jpg?color=red
     ```

   - 以下 CloudFront URL 用于使用备用域名和 HTTPS 协议的分配中的映像文件：

     ```
     https://www.example.com/images/image.jpg
     ```

2. **?**

   ? 表示查询字符串参数位于基本 URL 后面。即使您自己没有任何查询字符串参数，也请包含 ?。

3. ***###############*&**

   该值为可选项。如果您想添加自己的查询字符串参数，例如：

   ```
   color=red&size=medium
   ```

   则将该参数添加在 ? 之后，且位于 Expires 参数之前。在某些罕见情况下，可能需要将查询字符串参数放在 Key-Pair-Id 之后。

   > **Important**
   >
   > 您的参数不能命名为 Expires、Signature 或 Key-Pair-Id。

   如果您添加自己的参数，请在每个参数后附加 &，包括最后一个参数。

4. **Expires=*Unix ##################UTC#########***

   您希望 URL 不再允许访问文件的日期和时间。

   指定 Unix 时间格式（以秒为单位）和协调通用时间 (UTC) 格式的过期日期和时间。例如，UTC 时间 2013 年 1 月 1 日上午 10 点转换为 Unix 时间格式就是 1357034400，如本主题开头的示例所示。要使用纪元时间，请使用 32 位整数表示日期，该日期不得晚于 2147483647（2038 年 1 月 19 日，03:14:07 UTC）。有关 UTC 的信息，请参阅 RFC 3339，Internet 上的日期和时间：时间戳。

5. **&Signature=*###################***

   JSON 策略声明经过哈希处理、签署和 Base64 编码的版本。有关更多信息，请参阅 为使用标准策略的签名 URL 创建签名。

6. **&Key-Pair-Id=*CloudFront ######### ID#####################***

   CloudFront 公有密钥的 ID，例如，K2JCJMDEHXQW5F。公有密钥 ID 告诉 CloudFront 要使用哪个公有密钥来验证签名的 URL。CloudFront 将比较签名中的信息与策略声明中的信息，以确认该 URL 没有被篡改。

此公有密钥必须属于作为分配中可信签署人的密钥组。有关更多信息，请参阅 指定可以创建签名 URL 和签名 Cookie 的签署人。

为使用标准策略的签名 URL 创建签名

要为使用标准策略的签名 URL 创建签名，请完成以下步骤：

主题

- 为使用标准策略的签名 URL 创建策略声明
- 为使用标准策略的签名 URL 创建签名

为使用标准策略的签名 URL 创建策略声明

使用标准策略创建签名 URL 时，Signature 参数是策略声明经过哈希处理和签署的版本。对于使用标准策略的签名 URL，您没有像对待使用自定义策略的签名 URL 那样将策略声明包含在 URL 内。要创建策略声明，请执行以下过程。

为使用标准策略的签名 URL 创建策略声明

1. 使用以下 JSON 格式以及 UTF-8 字符编码构建策略声明。根据指定，准确包括所有标点符号和其他文本值。有关 Resource 和 DateLessThan 参数的信息，请参阅 在使用标准策略的签名 URL 的策略声明中指定的值。

```
{
    "Statement": [
        {
            "Resource": "base URL or stream name",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": ending date and time in Unix time format and
 UTC
                }
            }
        }
    ]
}
```

2. 删除策略声明中的所有空格（包括制表符和换行符）。您可能需要在应用程序代码的字符串中包括换码符。

在使用标准策略的签名 URL 的策略声明中指定的值

为标准策略创建策略声明时，请指定以下值。

资源

> **Note**
>
> 只能为 Resource 指定一个值。

包含查询字符串（如果有）的基本 URL，但不包括 CloudFront Expires、Signature 和 Key-Pair-Id 参数，例如：

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

请注意以下几点：

- 协议 – 该值必须以 http:// 或 https:// 开头。
- 查询字符串参数 – 如果没有查询字符串参数，请省略问号。
- 备用域名 – 如果在 URL 中指定备用域名 (CNAME)，则必须在引用网页或应用程序中的文件时指定备用域名。切勿为对象指定 Amazon S3 URL。

DateLessThan

Unix 时间格式（以秒为单位）和协调通用时间 (UTC) 格式的 URL 过期日期和时间。例如，2013 年 1 月 1 日上午 10 点 UTC 转换为 Unix 时间格式就是 1357034400。

该值必须与签名 URL 中的 Expires 查询字符串参数相匹配。切勿用引号将该值括起来。

有关更多信息，请参阅 CloudFront 何时检查签名 URL 中的过期日期和时间。

使用标准策略的签名 URL 的示例策略声明

当您在签名 URL 中使用以下示例策略声明时，用户将可以访问文件 https://d111111abcdef8.cloudfront.net/horizon.jpg，直至 UTC 时间 2013 年 1 月 1 日上午 10 点：

```
{
```

```
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/horizon.jpg?
size=large&license=yes",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
}
```

为使用标准策略的签名 URL 创建签名

要为签名 URL 中的 `Signature` 参数创建值，请对在为使用标准策略的签名 URL 创建策略声明中创建的策略声明进行哈希处理并签署。

有关额外信息以及如何哈希、签署及编码策略声明的示例，请参阅：

- 用于 Base64 编码和加密的 Linux 命令和 OpenSSL
- 为签名 URL 创建签名的代码示例

选项 1：使用标准策略创建签名

1. 使用 SHA-1 哈希函数和 RSA 对在为使用标准策略的签名 URL 创建策略声明过程中创建的策略声明进行哈希处理并签署。使用不再包含空格的策略声明版本。

   对于哈希函数所需的私有密钥，请使用其公有密钥位于分配的活动可信密钥组中的私有密钥。

   > **Note**
   >
   > 您用于哈希及签署策略声明的方法取决于您的编程语言和平台。有关代码示例，请参阅 为签名 URL 创建签名的代码示例。

2. 删除经过哈希处理并签署的字符串中的空格（包括制表符和换行符）。
3. 使用 MIME Base64 编码对字符串进行 Base64 编码。有关更多信息，请参阅 RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies 中的 Section 6.8, Base64 Content-Transfer-Encoding。
4. 用有效的字符替换 URL 查询字符串中的无效字符。下表列出了无效和有效字符。

替换这些无效字符	使用这些有效字符
+	- (连字符)
=	_（下划线）
/	~（波浪字符）

5. 将结果值附在签名 URL 的 &Signature= 之后，然后返回 要使用标准策略创建签名 URL，以完成签名 URL 的各部分的串连。

使用自定义策略创建签名 URL

要使用自定义策略创建签名 URL，请完成以下步骤。

要使用自定义策略创建签名 URL

1. 如果您使用 .NET 和 Java 创建签名 URL，而且，如果您尚未将密钥对私有密钥的格式从默认 .pem 格式重新设置为与 .NET 和 Java 兼容的格式，那么现在就开始设置吧。有关更多信息，请参阅 重新设置私有密钥的格式（仅限 .NET 和 Java）。
2. 按照列出的顺序连接以下值，复制本示例签名 URL 中显示的格式：

```
https://d111111abcdef8.cloudfront.net/
image.jpg?color=red&size=medium&Policy=eyANCiAgICEXAMPLEW1lbnQiOiBbeyANCiAgICAgICJSZXNvdXJj
j19DzZrvDh6hQ73lDx~-ar3UocvvRQVw6EkC~GdpGQyyOSKQim-
TxAnW7d8F5Kkai9HVx0FIu-5jcQb0UEmatEXAMPLE3ReXySpLSMj0yCd3ZAB4UcBCAqEijkytL6f3fVYNGQI6&Key-
Pair-Id=K2JCJMDEHXQW5F
```

删除所有空格（包括制表符和换行符）。您可能需要在应用程序代码的字符串中包括换码符。所有值的类型均为 String。

1. ***##### URL***

基本 URL 是您将用于访问文件的 CloudFront URL，如果您不使用签名 URL，包括您自己的查询字符串参数（如果有）。在上一示例中，基本 URL 为 https://d111111abcdef8.cloudfront.net/image.jpg。有关适用于分配的 URL 格式的更多信息，请参阅在 CloudFront 中自定义文件的 URL 格式。

以下示例显示了您为分配指定的值。

- 以下 CloudFront URL 适用于分配中的图像文件（使用 CloudFront 域名）。请注意，image.jpg 是在 images 目录中。URL 中文件的路径必须与您的 HTTP 服务器或 Amazon S3 存储桶中文件的路径匹配。

  ```
  https://d111111abcdef8.cloudfront.net/images/image.jpg
  ```

- 以下 CloudFront URL 包含查询字符串：

  ```
  https://d111111abcdef8.cloudfront.net/images/image.jpg?size=large
  ```

- 以下 CloudFront URL 适用于分配中的映像文件。两者都使用备用域名，第二个包括查询字符串：

  ```
  https://www.example.com/images/image.jpg
  ```

  ```
  https://www.example.com/images/image.jpg?color=red
  ```

- 以下 CloudFront URL 用于使用备用域名和 HTTPS 协议的分配中的映像文件：

  ```
  https://www.example.com/images/image.jpg
  ```

2. **?**

   ? 表示查询字符串参数位于基本 URL 后面。即使您自己没有任何查询字符串参数，也请包含 ?。

3. **###############&**

   该值为可选项。如果您想添加自己的查询字符串参数，例如：

   ```
   color=red&size=medium
   ```

   然后在 ? 之后和 Policy 参数之前添加它们。在某些罕见情况下，可能需要将查询字符串参数放在 Key-Pair-Id 之后。

   > **Important**
   >
   > 您的参数不能命名为 Policy、Signature 或 Key-Pair-Id。

   如果您添加自己的参数，请在每个参数后附加 &，包括最后一个参数。

4. **Policy=***##### Base64 ####*

   您的策略声明采用 JSON 格式，删除了空格，然后进行 Base64 编码。有关更多信息，请参阅为使用自定义策略的签名 URL 创建策略声明。

   策略语句控制签名 URL 授予用户的访问权限。它包括文件的 URL、过期日期和时间、URL 生效的可选日期和时间、允许访问文件的可选 IP 地址或 IP 地址范围。

5. **&Signature=***##################*

   JSON 策略声明经过哈希处理、签署和 Base64 编码的版本。有关更多信息，请参阅 为使用自定义策略的签名 URL 创建签名。

6. **&Key-Pair-Id=***CloudFront ######### ID######################*

   CloudFront 公有密钥的 ID，例如，K2JCJMDEHXQW5F。公有密钥 ID 告诉 CloudFront 要使用哪个公有密钥来验证签名的 URL。CloudFront 将比较签名中的信息与策略声明中的信息，以确认该 URL 没有被篡改。

   此公有密钥必须属于作为分配中可信签署人的密钥组。有关更多信息，请参阅 指定可以创建签名 URL 和签名 Cookie 的签署人。

为使用自定义策略的签名 URL 创建策略声明

要为使用自定义策略的签名 URL 创建策略声明，请完成以下步骤。

有关以各种方式控制访问文件的示例策略声明，请参阅the section called “使用自定义策略的签名 URL 的示例策略声明”。

为使用自定义策略的签名 URL 创建策略声明

1. 使用以下 JSON 格式构建策略声明。用自己的值替换小于 (<) 和大于 (>) 符号及其中的描述。有关更多信息，请参阅 the section called “在使用自定义策略的签名 URL 的策略声明中指定的值”。

```
{
    "Statement": [
        {
            "Resource": "<Optional but recommended: URL of the file>",
            "Condition": {
                "DateLessThan": {
                 "AWS:EpochTime": <Required: ending date and time in Unix time
 format and UTC>
```

```
                },
                "DateGreaterThan": {
                 "AWS:EpochTime": <Optional: beginning date and time in Unix time
 format and UTC>
                },
                "IpAddress": {
                 "AWS:SourceIp": "<Optional: IP address>"
                }
            }
        }
    ]
}
```

请注意以下几点：

- 您只能在策略中包含一个声明。
- 使用 UTF-8 字符编码。
- 根据指定，准确包括所有标点符号和参数名称。不接受参数名称的缩写。
- Condition 部分中参数的顺序无关紧要。
- 有关 Resource、DateLessThan、DateGreaterThan 和 IpAddress 值的信息，请参阅the section called “在使用自定义策略的签名 URL 的策略声明中指定的值”。

2. 删除策略声明中的所有空格（包括制表符和换行符）。您可能需要在应用程序代码的字符串中包括换码符。
3. 使用 MIME Base64 编码对策略声明进行 Base64 编码。有关更多信息，请参阅 RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies 中的 Section 6.8, Base64 Content-Transfer-Encoding。
4. 用有效的字符替换 URL 查询字符串中的无效字符。下表列出了无效和有效字符。

替换这些无效字符	使用这些有效字符
+	- (连字符)
=	_（下划线）
/	~（波浪字符）

5. 将结果值附在签名 URL Policy= 之后。

6. 通过对策略声明进行哈希、签署及 Base64 编码处理，创建签名 URL 的签名。有关更多信息，请参阅 the section called “为使用自定义策略的签名 URL 创建签名”。

在使用自定义策略的签名 URL 的策略声明中指定的值

针对自定义策略创建策略声明时，请指定以下值。

资源

包含任何查询字符串的 URL，但不包括 CloudFront `Policy`、`Signature` 和 `Key-Pair-Id` 参数。例如：

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg\?
size=large&license=yes
```

只能为 `Resource` 指定一个 URL 值。

> **Important**
>
> 您可以忽略策略中的 `Resource` 参数，但这样做意味着拥有签名 URL 任何人都可以访问与您用于创建签名 URL 的密钥对相关联的任何分配中的所有文件。

请注意以下几点：

- 协议 – 该值必须以 `http://`、`https://` 或 `*://` 开头。
- 查询字符串参数 – 如果 URL 有查询字符串参数，请使用反斜杠字符 (\) 来转义查询字符串开头的问号字符 (?)。例如：

  ```
  https://d111111abcdef8.cloudfront.net/images/horizon.jpg\?
  size=large&license=yes
  ```
- 通配符 – 您可以在策略的 URL 中使用通配符。支持以下通配符：
  - 星号 (*)，匹配零个或多个字符
  - 问号 (?)，正好匹配一个字符

  当 CloudFront 将策略中的 URL 与 HTTP 请求中的 URL 进行匹配时，策略中的 URL 分为四个部分：协议、域、路径和查询字符串，如下所示：

  ```
  [protocol]://[domain]/[path]\?[query string]
  ```

当您在策略的 URL 中使用通配符时，通配符匹配仅在包含该通配符的部分的界限内适用。例如，在策略中考虑此 URL:

`https://www.example.com/hello*world`

在此示例中，星号通配符 (*) 仅适用于路径部分，因此它与 URL `https://www.example.com/helloworld` 和 `https://www.example.com/hello-world` 匹配，但与 URL `https://www.example.net/hello?world` 不匹配。

以下例外适用于通配符匹配的部分界限：

- 路径部分中的尾部星号表示查询字符串部分中的星号。例如，`http://example.com/hello*` 等同于 `http://example.com/hello*\?*`。
- 域部分中的尾部星号表示路径部分和查询字符串部分都有星号。例如，`http://example.com*` 等同于 `http://example.com*/*\?*`。
- 策略中的 URL 可以忽略协议部分并在域部分中以星号开头。在这种情况下，协议部分被隐式设置为星号。例如，策略中的 URL `*example.com` 等同于 `*://*example.com/`。
- 星号本身 (`"Resource": "*"`) 与任何 URL 都匹配。

例如，策略中的值 `https://d111111abcdef8.cloudfront.net/*game_download.zip*` 与以下所有 URL 都匹配：

- `https://d111111abcdef8.cloudfront.net/game_download.zip`
- `https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes`
- `https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp`

- 备用域名 – 如果在策略的 URL 中指定备用域名 (CNAME)，则 HTTP 请求必须在网页或应用程序中使用该备用域名。请勿为策略中的文件指定 Amazon S3 URL。

DateLessThan

Unix 时间格式（以秒为单位）和协调通用时间 (UTC) 格式的 URL 过期日期和时间。在策略中，切勿用引号将该值括起来。有关 UTC 的信息，请参阅 Internet 上的日期和时间：时间戳。

例如，2023 年 1 月 31 日上午 10 点 UTC 转换为 Unix 时间格式就是 1675159200。

这是 `Condition` 部分唯一需要的参数。CloudFront 需要此值，以防止用户拥有对象的永久访问权。

有关更多信息，请参阅 the section called “CloudFront 何时检查签名 URL 中的过期日期和时间”

DateGreaterThan（可选）

Unix 时间格式（以秒为单位）和协调通用时间 (UTC) 格式的 URL 可选开始日期和时间。不允许用户在指定日期和时间或之前访问该文件。切勿用引号将该值括起来。

IpAddress（可选）

发出 HTTP 请求的客户端的 IP 地址。请注意以下几点：

- 要允许任何 IP 地址访问文件，请省略 `IpAddress` 参数。
- 可以指定一个 IP 地址或一个 IP 地址范围。如果客户端的 IP 地址在两个独立范围之一的范围内，您不能使用策略来允许访问。
- 要允许从单个 IP 地址访问，可指定：

  `"IPv4 IP ##/32"`
- 必须采用标准 IPv4 CIDR 格式指定 IP 地址范围（例如，`192.0.2.0/24`）。有关更多信息，请参阅 Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan。

  > **Important**
  >
  > 不支持 IPv6 格式的 IP 地址，如 2001:0db8:85a3::8a2e:0370:7334。

  如果使用包含 `IpAddress` 的自定义策略，请勿为分配启用 IPv6。如果希望通过 IP 地址限制对某些内容的访问并支持其他内容的 IPv6 请求，可以创建两个分配。有关更多信息，请参阅 the section called “启用 IPv6”主题中的 the section called “分配设置”。

使用自定义策略的签名 URL 的示例策略声明

以下示例策略声明显示了如何控制对特定文件、目录中的所有文件或与密钥对 ID 有关的所有文件的访问。这些示例也显示了如何控制来自单个 IP 地址或 IP 地址范围的访问，以及如何防止用户在指定日期和时间后使用签名 URL。

如果复制并粘贴其中的任何示例，请删除任何空格（包括制表符和换行符），将值替换为自己的值，并在右大括号（}）后面包含一个换行符。

有关更多信息，请参阅 the section called “在使用自定义策略的签名 URL 的策略声明中指定的值”。

主题

示例策略声明：从 IP 地址范围访问一个文件

签名 URL 中的以下示例自定义策略指定用户可从范围 192.0.2.0/24 内的 IP 地址访问文件 https://d111111abcdef8.cloudfront.net/game_download.zip，直至 UTC 时间 2023 年 1 月 31 日上午 10 点：

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/game_download.zip",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675159200
                }
            }
        }
    ]
}
```

示例策略声明：从 IP 地址范围访问一个目录中的所有文件

以下示例自定义策略允许您为 training 目录中的任何文件创建签名 URL，如 Resource 参数中的星号通配符 (*) 所指示。用户可从范围 192.0.2.0/24 内的 IP 地址访问文件，直至 UTC 时间 2023 年 1 月 31 日上午 10 点：

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/training/*",
            "Condition": {
                "IpAddress": {
```

```
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675159200
                }
            }
        }
    ]
}
```

您在其中使用此策略的每个签名 URL 包括确定特定文件的 URL，例如：

`https://d111111abcdef8.cloudfront.net/training/orientation.pdf`

示例策略声明：从一个 IP 地址访问与一个密钥对 ID 关联的所有文件

以下示例自定义策略允许您为与任何分配有关的任何文件创建签名 URL，如 `Resource` 参数中的星号通配符 (*) 所指示。签名 URL 必须使用 `https://` 协议，而不是 `http://`。用户必须使用 IP 地址 `192.0.2.10/32`。（CIDR 表示法中的值 `192.0.2.10/32` 指代单个 IP 地址 `192.0.2.10`。）这些文件仅从 UTC 时间 2023 年 1 月 31 日上午 10 点到 UTC 时间 2023 年 2 月 2 日上午 10 点期间可用：

```
{
    "Statement": [
       {
            "Resource": "https://*",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.10/32"
                },
                "DateGreaterThan": {
                    "AWS:EpochTime": 1675159200
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1675332000
                }
            }
        }
    ]
}
```

您在其中使用此策略的每个签名 URL 包括确定特定 CloudFront 分配中特定文件的 URL，例如：

```
https://d111111abcdef8.cloudfront.net/training/orientation.pdf
```

签名 URL 还包括密钥对 ID，它必须与您在 URL 中指定的分配 (d111111abcdef8.cloudfront.net) 中的可信密钥组关联。

为使用自定义策略的签名 URL 创建签名

使用自定义策略的签名 URL 的签名是策略声明的哈希、签署及 Base64 编码版本。要为自定义策略创建签名，请完成以下步骤。

有关额外信息以及如何哈希、签署及编码策略声明的示例，请参阅：

- 用于 Base64 编码和加密的 Linux 命令和 OpenSSL
- 为签名 URL 创建签名的代码示例

选项 1：使用自定义策略创建签名

1. 使用 SHA-1 哈希函数和 RSA 对在为使用自定义策略的签名 URL 创建策略声明过程中创建的 JSON 策略声明进行哈希处理并签署。使用不再包含空格但尚未进行 Base64 编码的策略声明版本。

   对于哈希函数所需的私有密钥，请使用其公有密钥位于分配的活动可信密钥组中的私有密钥。

   > **Note**
   >
   > 您用于哈希及签署策略声明的方法取决于您的编程语言和平台。有关代码示例，请参阅为签名 URL 创建签名的代码示例。

2. 删除经过哈希处理并签署的字符串中的空格（包括制表符和换行符）。
3. 使用 MIME Base64 编码对字符串进行 Base64 编码。有关更多信息，请参阅 RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies 中的 Section 6.8, Base64 Content-Transfer-Encoding。
4. 用有效的字符替换 URL 查询字符串中的无效字符。下表列出了无效和有效字符。

替换这些无效字符	使用这些有效字符
+	- (连字符)
=	_（下划线）

替换这些无效字符	使用这些有效字符
/	~（波浪字符）

5. 将结果值附在签名 URL 的 `&Signature=` 之后，然后返回 要使用自定义策略创建签名 URL，以完成签名 URL 的各部分的串连。

使用签名 Cookie

当您不希望更改当前 URL，或者希望提供对多个限制文件（例如，网站订户区域中的所有文件）的访问权时，可使用 CloudFront 签名 Cookie 控制能够访问内容的人员。本主题介绍了使用签名 Cookie 时的注意事项，并介绍了如何使用标准策略和自定义策略设置签名 Cookie。

主题

- 决定为签名 Cookie 使用标准策略或自定义策略
- 签名 Cookie 的工作方式
- 防止滥用签名 Cookie
- CloudFront 何时检查签名 Cookie 中的过期日期和时间
- 代码示例和第三方工具
- 使用标准策略设置签名 Cookie
- 使用自定义策略设置签名 Cookie

决定为签名 Cookie 使用标准策略或自定义策略

创建签名 Cookie 时，以 JSON 格式编写指定签名 Cookie 限制的策略声明，例如，URL 的有效期。可以使用标准策略或自定义策略。下表比较了标准策略和自定义策略：

说明	标准策略	自定义策略
可对多个文件重复使用策略声明。要重复使用策略声明，您必须在 `Resource` 对象中使用通配符。有关更多信息，请参阅在使用自定义策略的签名 Cookie 的策略声明中指定的值。	否	是
可指定用户开始访问内容的日期和时间	否	是（可选）

说明	标准策略	自定义策略
可指定用户无法再访问内容的日期和时间	是	是
可指定能够访问内容的用户的 IP 地址或 IP 地址范围	否	是（可选）

有关使用标准策略创建签名 Cookie 的信息，请参阅使用标准策略设置签名 Cookie。

有关使用自定义策略创建签名 Cookie 的信息，请参阅使用自定义策略设置签名 Cookie。

签名 Cookie 的工作方式

下面概述了如何为签名 Cookie 配置 CloudFront，并概述了当用户提交包含签名 Cookie 的请求时 CloudFront 如何响应。

1. 在 CloudFront 分配中，指定一个或多个可信密钥组，这些密钥组包含 CloudFront 可用来验证 URL 签名的公有密钥。您可以使用相应的私有密钥对 URL 进行签名。

   有关更多信息，请参阅 指定可以创建签名 URL 和签名 Cookie 的签署人。
2. 您开发应用程序，以确定用户是否应对您的内容拥有访问权限，如果是，则向查看器发送三个 `Set-Cookie` 标头。（每个 `Set-Cookie` 标头只能包含一个名称-值对，CloudFront 签名 Cookie 需要三个名称-值对。）必须先向查看器发送 `Set-Cookie` 标头，查看器才能请求私有内容。如果对 Cookie 设置一个较短的过期时间，您可能还需要发送另外三个 `Set-Cookie` 标头以响应后续请求，从而确保用户继续拥有访问权限。

   通常情况下，CloudFront 分配至少拥有两个缓存行为，一个不需要身份验证，另一个则需要。站点安全部分的错误页面包含一个指向登录页面的重定向器或链接。

   如果将分配配置为根据 Cookie 缓存文件，CloudFront 则不会基于签名 Cookie 中的属性缓存单独的文件。
3. 用户登录您的网站，为内容付费或满足一些其他访问要求。
4. 应用程序在响应中返回 `Set-Cookie` 标头，查看器存储名称-值对。
5. 用户请求文件。

   用户的浏览器或其他查看器获取第 4 步中的名称-值对，并将它们添加到请求的 `Cookie` 标头中。这就是签名 Cookie。
6. CloudFront 使用公有密钥验证签名 Cookie 中的签名并确认该 Cookie 未被篡改。如果签名无效，则请求将被拒绝。

如果 Cookie 中的签名有效，CloudFront 将查看 Cookie 中的策略声明（如果使用标准策略，则构造一个），以确认该请求仍然有效。例如，如果为 Cookie 指定了开始和结束日期及时间，CloudFront 会确认用户是否是在您希望允许访问的时间段尝试访问内容。

如果请求满足策略声明中的要求，CloudFront 将像提供不受限制的内容那样提供内容：确定文件是否已在边缘缓存中，必要时将请求转发到源，然后向用户返回文件。

防止滥用签名 Cookie

如果您在 `Domain` 标头中指定 `Set-Cookie` 参数，请尽可能指定最精确的值，以减少具有相同根域名的某些人的潜在访问。例如，app.example.com 优于 example.com，尤其是在 example.com 不由您控制时。这有助于防止他人从 www.example.com 访问内容。

为帮助防止此类攻击，请执行以下操作：

- 排除 `Expires` 和 `Max-Age` Cookie 属性，以便 `Set-Cookie` 标头创建会话 Cookie。当用户关闭浏览器时，系统会自动删除会话 Cookie，这降低了其他人非法访问内容的可能性。
- 包括 `Secure` 属性，以便在查看器将 Cookie 包含在请求中时对 Cookie 进行加密。
- 如果可能，请使用自定义策略并包含查看器的 IP 地址。
- 在 `CloudFront-Expires` 属性中，根据您希望用户对内容拥有访问权限的时间长度指定最短合理过期时间。

CloudFront 何时检查签名 Cookie 中的过期日期和时间

为了确定签名 Cookie 是否仍然有效，CloudFront 会在发出 HTTP 请求时检查 Cookie 中的过期日期和时间。如果客户端刚好在过期时间之前开始下载大型文件，即使在下载过程中到了过期时间，该下载也应该完成。如果 TCP 连接断开，并且客户端试图在过期时间到期后重新开始下载，则下载将会失败。

如果客户端使用 Range GET 来获取较小的文件，在过期时间到期后发生的任何 GET 请求将会失败。有关 Range GET 的更多信息，请参阅 CloudFront 如何处理对象的部分请求 (Range GET)。。

代码示例和第三方工具

私有内容的代码示例仅显示如何为签名 URL 创建签名。不过，为签名 Cookie 创建签名的过程非常相似，因此代码示例中的很多内容仍然有用。有关更多信息，请参阅以下主题：

- 使用 Perl 创建 URL 签名
- 使用 PHP 创建 URL 签名

- 使用 C# 和 .NET Framework 创建 URL 签名
- 使用 Java 创建 URL 签名

使用标准策略设置签名 Cookie

要使用标准策略设置签名 Cookie，请完成以下步骤。要创建签名，请参阅为使用标准策略的签名 Cookie 创建签名。

使用标准策略设置签名 Cookie

1. 如果您要使用 .NET 和 Java 创建签名 Cookie，但尚未将密钥对私有密钥的格式从默认 .pem 格式重新设置为与 .NET 和 Java 兼容的格式，请立即执行该操作。有关更多信息，请参阅 重新设置私有密钥的格式（仅限 .NET 和 Java）。
2. 将应用程序编程为向批准的查看器发送三个 `Set-Cookie` 标头。您之所以需要三个 `Set-Cookie` 标头，是因为每个 `Set-Cookie` 标头只能包含一个名称-值对，而 CloudFront 签名 Cookie 需要三个名称-值对。名称值对为：`CloudFront-Expires`、`CloudFront-Signature` 和 `CloudFront-Key-Pair-Id`。在用户首次请求您希望控制访问权的文件时，查看器中必须包含这些值。

   > **Note**
   >
   > 一般情况下，建议排除 `Expires` 和 `Max-Age` 属性。如果排除这些属性，那么当用户关闭浏览器时，浏览器会删除 Cookie，这降低了其他人非法访问内容的可能性。有关更多信息，请参阅 防止滥用签名 Cookie。

   Cookie 属性的名称区分大小写。

   包含换行符的目的只是为了让属性的可读性更佳。

```
Set-Cookie:
CloudFront-Expires=date and time in Unix time format (in seconds) and Coordinated
 Universal Time (UTC);
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
```

```
CloudFront-Signature=hashed and signed version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
CloudFront-Key-Pair-Id=public key ID for the CloudFront public key whose
 corresponding private key you're using to generate the signature;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly
```

（可选）**Domain**

所请求文件的域名。如果不指定 Domain 属性，则默认值是 URL 中的域名，它仅适用于指定的域名，而不适用于子域。如果指定 Domain 属性，它还将适用于子域。可以选择在域名前加一个句点（例如 Domain=.example.com）。此外，如果指定 Domain 属性，那么 URL 中的域名则必须与 Domain 属性的值匹配。

可以指定 CloudFront 为您的分配指定的域名（例如 d111111abcdef8.cloudfront.net），但不能为域名指定 *.cloudfront.net。

如果希望在 URL 中使用备用域名（如 example.com），则必须向分配添加备用域名，而无论是否指定 Domain 属性。有关更多信息，请参阅 备用域名 (CNAME)主题中的 分配设置参考。

（可选）**Path**

所请求文件的路径。如果未指定 Path 属性，则默认值是 URL 中的路径。

Secure

要求查看器先加密 Cookie，然后再发送请求。建议通过 HTTPS 连接发送 Set-Cookie 标头，以确保 Cookie 受到保护，而不会受到中间人攻击。

HttpOnly

定义浏览器（如果受支持）如何与 Cookie 值进行交互。如果选择 HttpOnly，则 JavaScript 无法访问 Cookie 值。这种预防措施有助于缓解跨站点脚本（XSS）攻击。有关更多信息，请参阅使用 HTTP Cookie。

`CloudFront-Expires`

指定 Unix 时间格式（以秒为单位）和协调通用时间 (UTC) 格式的过期日期和时间。例如，2013 年 1 月 1 日上午 10 点 UTC 转换为 Unix 时间格式就是 1357034400。要使用纪元时间，请使用 32 位整数表示日期，该日期不得晚于 2147483647（2038 年 1 月 19 日，03:14:07 UTC）。有关 UTC 的信息，请参阅 RFC 3339，Internet 上的日期和时间：时间戳，网址为 https://tools.ietf.org/html/rfc3339。

`CloudFront-Signature`

JSON 策略声明经过哈希处理、签署和 Base64 编码的版本。有关更多信息，请参阅 为使用标准策略的签名 Cookie 创建签名。

`CloudFront-Key-Pair-Id`

CloudFront 公有密钥的 ID，例如，`K2JCJMDEHXQW5F`。公有密钥 ID 告诉 CloudFront 要使用哪个公有密钥来验证签名的 URL。CloudFront 将比较签名中的信息与策略声明中的信息，以确认该 URL 没有被篡改。

此公有密钥必须属于作为分配中可信签署人的密钥组。有关更多信息，请参阅 指定可以创建签名 URL 和签名 Cookie 的签署人。

以下示例显示了在文件的 URL 中使用与分配关联的域名时一个签名 Cookie 的 `Set-Cookie` 标头：

```
Set-Cookie: CloudFront-Expires=1426500000; Domain=d111111abcdef8.cloudfront.net; Path=/
images/*; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=yXrSIgyQoeE4FBI4eMKF6ho~CA8_;
 Domain=d111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=d111111abcdef8.cloudfront.net; Path=/images/*; Secure; HttpOnly
```

以下示例显示了在文件的 URL 中使用备用域名 example.org 时一个签名 Cookie 的 `Set-Cookie` 标头：

```
Set-Cookie: CloudFront-Expires=1426500000; Domain=example.org; Path=/images/*; Secure;
 HttpOnly
Set-Cookie: CloudFront-Signature=yXrSIgyQoeE4FBI4eMKF6ho~CA8_; Domain=example.org;
 Path=/images/*; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/images/*;
 Secure; HttpOnly
```

如果希望在 URL 中使用备用域名（如 example.com），则必须向分配添加备用域名，而无论是否指定 Domain 属性。有关更多信息，请参阅 备用域名 (CNAME)主题中的 分配设置参考。

为使用标准策略的签名 Cookie 创建签名

要为使用标准策略的签名 Cookie 创建签名，请完成以下步骤。

主题

- 为使用标准策略的签名 Cookie 创建策略声明
- 签署策略声明，以便为使用标准策略的签名 Cookie 创建签名

为使用标准策略的签名 Cookie 创建策略声明

设置使用标准策略的签名 Cookie 时，CloudFront-Signature 属性是策略声明的经过哈希处理和签署的版本。对于使用标准策略的签名 Cookie，请不要像在使用自定义策略的签名 Cookie 一样在 Set-Cookie 标头中包含策略声明。要创建策略语句，请完成以下步骤。

为使用标准策略的签名 Cookie 创建策略声明

1. 使用以下 JSON 格式以及 UTF-8 字符编码构建策略声明。根据指定，准确包括所有标点符号和其他文本值。有关 Resource 和 DateLessThan 参数的信息，请参阅 在策略声明中为使用标准策略的签名 Cookie 指定的值。

```
{
    "Statement": [
        {
            "Resource": "base URL or stream name",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": ending date and time in Unix time format and
 UTC
                }
            }
        }
    ]
}
```

2. 删除策略声明中的所有空格（包括制表符和换行符）。您可能需要在应用程序代码的字符串中包括换码符。

在策略声明中为使用标准策略的签名 Cookie 指定的值

为标准策略创建策略声明时，请指定以下值：

资源

包含查询字符串 (如果有) 的基本 URL，例如：

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

只能为 Resource 指定一个值。

请注意以下几点：

- 协议 – 该值必须以 http:// 或 https:// 开头。
- 查询字符串参数 – 如果没有查询字符串参数，请省略问号。
- 备用域名 – 如果在 URL 中指定备用域名 (CNAME)，则必须在引用网页或应用程序中的文件时指定备用域名。请勿为文件指定 Amazon S3 URL。

DateLessThan

Unix 时间格式（以秒为单位）和协调通用时间 (UTC) 格式的 URL 过期日期和时间。切勿用引号将该值括起来。

例如，UTC 时间 2015 年 3 月 16 日上午 10 点转换为 Unix 时间格式就是 1426500000。

该值必须与 CloudFront-Expires 标头中的 Set-Cookie 属性的值匹配。切勿用引号将该值括起来。

有关更多信息，请参阅 CloudFront 何时检查签名 Cookie 中的过期日期和时间。

标准策略的示例策略声明

当在签名 Cookie 中使用以下示例策略声明时，用户可访问文件 https://d111111abcdef8.cloudfront.net/horizon.jpg，直至 UTC 时间 2015 年 3 月 16 日上午 10 点：

```
{
    "Statement": [
```

```
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/horizon.jpg?
size=large&license=yes",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime": 1426500000
                }
            }
        }
    ]
}
```

签署策略声明，以便为使用标准策略的签名 Cookie 创建签名

要为 CloudFront-Signature 标头中的 Set-Cookie 属性创建值，请对在 为使用标准策略的签名 Cookie 创建策略声明 中创建的策略声明进行哈希处理并签署。

有关如何对策略声明进行哈希、签署及编码的更多信息和示例，请参阅以下主题：

- 用于 Base64 编码和加密的 Linux 命令和 OpenSSL
- 为签名 URL 创建签名的代码示例

为使用标准策略的签名 Cookie 创建签名

1. 使用 SHA-1 哈希函数和 RSA 对在为使用标准策略的签名 Cookie 创建策略声明过程中创建的策略声明进行哈希处理并签署。使用不再包含空格的策略声明版本。

   对于哈希函数所需的私有密钥，请使用其公有密钥位于分配的活动可信密钥组中的私有密钥。

   > **Note**
   >
   > 您用于哈希及签署策略声明的方法取决于您的编程语言和平台。有关代码示例，请参阅 为签名 URL 创建签名的代码示例。

2. 删除经过哈希处理并签署的字符串中的空格（包括制表符和换行符）。
3. 使用 MIME Base64 编码对字符串进行 Base64 编码。有关更多信息，请参阅 RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies 中的 Section 6.8, Base64 Content-Transfer-Encoding。
4. 用有效的字符替换 URL 查询字符串中的无效字符。下表列出了无效和有效字符。

替换这些无效字符	使用这些有效字符
+	- (连字符)
=	_ (下划线)
/	~ (波浪字符)

5. 在 Set-Cookie 名称-值对的 CloudFront-Signature 标头中包含结果值。然后返回到使用标准策略设置签名 Cookie为 CloudFront-Key-Pair-Id 添加 Set-Cookie 标头。

使用自定义策略设置签名 Cookie

要使用自定义策略设置签名 Cookie，请完成以下步骤。

使用自定义策略设置签名 Cookie

1. 如果您使用 .NET 和 Java 创建签名 URL，而且，如果您尚未将密钥对私有密钥的格式从默认 .pem 格式重新设置为与 .NET 和 Java 兼容的格式，那么现在就开始设置吧。有关更多信息，请参阅 重新设置私有密钥的格式 (仅限 .NET 和 Java)。
2. 将应用程序编程为向批准的查看器发送三个 Set-Cookie 标头。您之所以需要三个 Set-Cookie 标头，是因为每个 Set-Cookie 标头只能包含一个名称-值对，而 CloudFront 签名 Cookie 需要三个名称-值对。名称值对为：CloudFront-Policy、CloudFront-Signature 和 CloudFront-Key-Pair-Id。在用户首次请求您希望控制访问权的文件时，查看器中必须包含这些值。

   > **Note**
   >
   > 一般情况下，建议排除 Expires 和 Max-Age 属性。如果排除这些属性，那么当用户关闭浏览器时，浏览器则会删除 Cookie，这降低了其他人非法访问内容的可能性。有关更多信息，请参阅 防止滥用签名 Cookie。

   Cookie 属性的名称区分大小写。

   包含换行符的目的只是为了让属性的可读性更佳。

```
Set-Cookie:
```

```
CloudFront-Policy=base64 encoded version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly


Set-Cookie:
CloudFront-Signature=hashed and signed version of the policy statement;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly

Set-Cookie:
CloudFront-Key-Pair-Id=public key ID for the CloudFront public key whose
 corresponding private key you're using to generate the signature;
Domain=optional domain name;
Path=/optional directory path;
Secure;
HttpOnly
```

（可选）**Domain**

所请求文件的域名。如果不指定 Domain 属性，则默认值是 URL 中的域名，它仅适用于指定的域名，而不适用于子域。如果指定 Domain 属性，它还将适用于子域。可以选择在域名前加一个句点（例如 Domain=.example.com）。此外，如果指定 Domain 属性，那么 URL 中的域名则必须与 Domain 属性的值匹配。

可以指定 CloudFront 为您的分配指定的域名（例如 d111111abcdef8.cloudfront.net），但不能为域名指定 *.cloudfront.net。

如果希望在 URL 中使用备用域名（如 example.com），则必须向分配添加备用域名，而无论是否指定 Domain 属性。有关更多信息，请参阅 备用域名 (CNAME)主题中的 分配设置参考。

（可选）**Path**

所请求文件的路径。如果未指定 Path 属性，则默认值是 URL 中的路径。

Secure

要求查看器先加密 Cookie，然后再发送请求。建议通过 HTTPS 连接发送 Set-Cookie 标头，以确保 Cookie 受到保护，而不会受到中间人攻击。

`HttpOnly`

要求查看器仅在 HTTP 或 HTTPS 请求中发送 Cookie。

`CloudFront-Policy`

您的策略声明采用 JSON 格式，删除了空格，然后进行 Base64 编码。有关更多信息，请参阅为使用自定义策略的签名 Cookie 创建签名。

策略语句控制签名 Cookie 授予用户的访问权限。它包括用户可以访问的文件、过期日期和时间、URL 生效的可选日期和时间、允许访问文件的可选 IP 地址或 IP 地址范围。

`CloudFront-Signature`

JSON 策略声明经过哈希处理、签署和 Base64 编码的版本。有关更多信息，请参阅为使用自定义策略的签名 Cookie 创建签名。

`CloudFront-Key-Pair-Id`

CloudFront 公有密钥的 ID，例如，`K2JCJMDEHXQW5F`。公有密钥 ID 告诉 CloudFront 要使用哪个公有密钥来验证签名的 URL。CloudFront 将比较签名中的信息与策略声明中的信息，以确认该 URL 没有被篡改。

此公有密钥必须属于作为分配中可信签署人的密钥组。有关更多信息，请参阅指定可以创建签名 URL 和签名 Cookie 的签署人。

自定义策略的 **`Set-Cookie`** 标头示例

请参阅以下 `Set-Cookie` 标头对示例。

如果您希望在 URL 中使用备用域名（如 example.com），则无论您是否指定 `Domain` 属性，都必须向分配中添加备用域名。有关更多信息，请参阅 备用域名 (CNAME)主题中的 分配设置参考。

Example 示例 1

在文件的 URL 中使用与分配关联的域名时，您可以将 `Set-Cookie` 标头用于一个已签名的 Cookie。

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
```

```
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
```

Example 示例 2

在文件的 URL 中使用备用域名（example.org）时，您可以将 Set-Cookie 标头用于一个已签名的 Cookie。

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=example.org; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_; Domain=example.org;
 Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/; Secure;
 HttpOnly
```

Example 示例 3

在文件的 URL 中使用与分配关联的域名时，您可以将 Set-Cookie 标头对用于一个已签名的请求。

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_;
 Domain=d111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F;
 Domain=dd111111abcdef8.cloudfront.net; Path=/; Secure; HttpOnly
```

Example 示例 4

在文件的 URL 中使用与分配关联的备用域名（example.org）时，您可以将 Set-Cookie 标头对用于一个已签名的请求。

```
Set-Cookie: CloudFront-
Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2QxMTExMTFhYmNkZWY4LmNsb3VkZnJvbnQubmV0L2dh
 Domain=example.org; Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Signature=dtKhpJ3aUYxqDIwepczPiDb9NXQ_; Domain=example.org;
 Path=/; Secure; HttpOnly
Set-Cookie: CloudFront-Key-Pair-Id=K2JCJMDEHXQW5F; Domain=example.org; Path=/; Secure;
 HttpOnly
```

为使用自定义策略的签名 Cookie 创建策略声明

要创建自定义策略的策略声明，请完成以下步骤。有关以各种方式控制访问文件的一些示例策略声明，请参阅使用自定义策略的签名 Cookie 的示例策略声明。

为使用自定义策略的签名 Cookie 创建策略声明

1. 使用以下 JSON 格式构建策略声明。

```
{
    "Statement": [
        {
            "Resource": "URL of the file",
            "Condition": {
                "DateLessThan": {
                    "AWS:EpochTime":required ending date and time in Unix time
 format and UTC
                },
                "DateGreaterThan": {
                    "AWS:EpochTime":optional beginning date and time in Unix time
 format and UTC
                },
                "IpAddress": {
                    "AWS:SourceIp": "optional IP address"
                }
            }
        }
    ]
}
```

   请注意以下几点：

   - 您只能包含一个语句。
   - 使用 UTF-8 字符编码。
   - 根据指定，准确包括所有标点符号和参数名称。不接受参数名称的缩写。
   - Condition 部分中参数的顺序无关紧要。
   - 有关 Resource、DateLessThan、DateGreaterThan 和 IpAddress 值的信息，请参阅在使用自定义策略的签名 Cookie 的策略声明中指定的值。
2. 删除策略声明中的所有空格（包括制表符和换行符）。您可能需要在应用程序代码的字符串中包括换码符。

3. 使用 MIME Base64 编码对策略声明进行 Base64 编码。有关更多信息，请参阅 RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies 中的 Section 6.8, Base64 Content-Transfer-Encoding。
4. 用有效的字符替换 URL 查询字符串中的无效字符。下表列出了无效和有效字符。

替换这些无效字符	使用这些有效字符
+	- (连字符)
=	_（下划线）
/	~（波浪字符）

5. 将结果值包含在 `Set-Cookie` 标头的 `CloudFront-Policy=` 之后。
6. 通过对策略声明进行哈希、签署及 Base64 编码处理，为 `CloudFront-Signature` 的 `Set-Cookie` 标头创建签名。有关更多信息，请参阅 为使用自定义策略的签名 Cookie 创建签名。

在使用自定义策略的签名 Cookie 的策略声明中指定的值

针对自定义策略创建策略声明时，请指定以下值。

资源

包含查询字符串 (如果有) 的基本 URL：

```
https://d111111abcdef8.cloudfront.net/images/horizon.jpg?
size=large&license=yes
```

> **Important**
>
> 如果省略 `Resource` 参数，用户将可以访问与用于创建签名 URL 的密钥对相关联的任何分配的所有关联文件。

只能为 `Resource` 指定一个值。

请注意以下几点：

- 协议 – 该值必须以 `http://` 或 `https://` 开头。
- 查询字符串参数 – 如果没有查询字符串参数，请省略问号。
- 通配符 – 可以使用与零个或更多个字符匹配的通配符 (*)，或者与字符串中的任何位置的一个字符完全匹配的通配符 (?)。例如，值：

  ```
  https://d111111abcdef8.cloudfront.net/*game_download.zip*
  ```

  将包括（例如）以下文件：

  - `https://d111111abcdef8.cloudfront.net/game_download.zip`
  - `https://d111111abcdef8.cloudfront.net/example_game_download.zip?license=yes`
  - `https://d111111abcdef8.cloudfront.net/test_game_download.zip?license=temp`
- 备用域名 – 如果在 URL 中指定备用域名 (CNAME)，则必须在引用网页或应用程序中的文件时指定备用域名。请勿为文件指定 Amazon S3 URL。

DateLessThan

Unix 时间格式（以秒为单位）和协调通用时间 (UTC) 格式的 URL 过期日期和时间。切勿用引号将该值括起来。

例如，UTC 时间 2015 年 3 月 16 日上午 10 点转换为 Unix 时间格式就是 1426500000。

有关更多信息，请参阅 CloudFront 何时检查签名 Cookie 中的过期日期和时间。

DateGreaterThan（可选）

Unix 时间格式（以秒为单位）和协调通用时间 (UTC) 格式的 URL 可选开始日期和时间。不允许用户在指定日期和时间或之前访问该文件。切勿用引号将该值括起来。

IpAddress（可选）

发出 GET 请求的客户端的 IP 地址。请注意以下几点：

- 要允许任何 IP 地址访问文件，请省略 `IpAddress` 参数。
- 可以指定一个 IP 地址或一个 IP 地址范围。例如，如果客户端的 IP 地址在两个独立范围之一的范围内，您不能设置策略以允许访问。
- 要允许从单个 IP 地址访问，可指定：

  ```
  "IPv4 IP ##/32"
  ```

- 必须采用标准 IPv4 CIDR 格式指定 IP 地址范围 (例如，`192.0.2.0/24`)。有关更多信息，请转到《RFC 4632，无类别域间路由 (CIDR)：互联网地址分配和聚合计划》，网址为 https://tools.ietf.org/html/rfc4632。

  > **Important**
  >
  > 不支持 IPv6 格式的 IP 地址，如 2001:0db8:85a3::8a2e:0370:7334。

  如果使用包含 `IpAddress` 的自定义策略，请勿为分配启用 IPv6。如果希望通过 IP 地址限制对某些内容的访问并支持其他内容的 IPv6 请求，可以创建两个分配。有关更多信息，请参阅 启用 IPv6主题中的 分配设置参考。

使用自定义策略的签名 Cookie 的示例策略声明

以下示例策略声明显示了如何控制对特定文件、目录中的所有文件或与密钥对 ID 有关的所有文件的访问。这些示例也显示了如何控制来自单个 IP 地址或 IP 地址范围的访问，以及如何防止用户在指定日期和时间后使用签名 Cookie。

如果复制并粘贴其中的任何示例，请删除任何空格（包括制表符和换行符），将值替换为自己的值，并在右大括号 (}) 后面包含一个换行符。

有关更多信息，请参阅 在使用自定义策略的签名 Cookie 的策略声明中指定的值。

主题

- 示例策略声明：从 IP 地址范围访问一个文件
- 示例策略声明：从 IP 地址范围访问一个目录中的所有文件
- 示例策略声明：从一个 IP 地址访问与一个密钥对 ID 关联的所有文件

示例策略声明：从 IP 地址范围访问一个文件

签名 Cookie 中的以下示例自定义策略指定用户可从范围 `192.0.2.0/24` 内的 IP 地址访问文件 `https://d111111abcdef8.cloudfront.net/game_download.zip`，直至 UTC 时间 2023 年 1 月 1 日上午 10 点：

```
{
    "Statement": [
        {
```

```
            "Resource": "https://d111111abcdef8.cloudfront.net/game_download.zip",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
}
```

示例策略声明：从 IP 地址范围访问一个目录中的所有文件

以下示例自定义策略允许为 `training` 目录中的任何文件创建签名 Cookie，如 `Resource` 参数中的 * 通配符所指示。用户可从范围 `192.0.2.0/24` 内的 IP 地址访问文件，直至 UTC 时间 2013 年 1 月 1 日上午 10 点：

```
{
    "Statement": [
        {
            "Resource": "https://d111111abcdef8.cloudfront.net/training/*",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.0/24"
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357034400
                }
            }
        }
    ]
}
```

在其中使用此策略的每个签名 Cookie 包括用于标识特定文件的基本 URL，例如：

`https://d111111abcdef8.cloudfront.net/training/orientation.pdf`

示例策略声明：从一个 IP 地址访问与一个密钥对 ID 关联的所有文件

以下示例自定义策略允许为与任何分配相关联的任何文件设置签名 Cookie，如 `Resource` 参数中的 * 通配符所指示。用户必须使用 IP 地址 `192.0.2.10/32`。（CIDR 表示法中的值 `192.0.2.10/32` 指

代单个 IP 地址 192.0.2.10。）这些文件仅从 UTC 时间 2013 年 1 月 1 日上午 10 点到 UTC 时间 2013 年 1 月 2 日上午 10 点期间可用：

```
{
    "Statement": [
        {
            "Resource": "https://*",
            "Condition": {
                "IpAddress": {
                    "AWS:SourceIp": "192.0.2.10/32"
                },
                "DateGreaterThan": {
                    "AWS:EpochTime": 1357034400
                },
                "DateLessThan": {
                    "AWS:EpochTime": 1357120800
                }
            }
        }
    ]
}
```

在其中使用此策略的每个签名 Cookie 包括用于标识特定 CloudFront 分配中特定文件的基本 URL，例如：

`https://d111111abcdef8.cloudfront.net/training/orientation.pdf`

签名 Cookie 还包括密钥对 ID，它必须与您在基本 URL 中指定的分配 (d111111abcdef8.cloudfront.net) 中的可信密钥组关联。

为使用自定义策略的签名 Cookie 创建签名

使用自定义策略的签名 Cookie 的签名是策略声明的哈希、签署及 Base64 编码版本。

有关额外信息以及如何哈希、签署及编码策略声明的示例，请参阅：

- 用于 Base64 编码和加密的 Linux 命令和 OpenSSL
- 为签名 URL 创建签名的代码示例

为使用自定义策略的签名 Cookie 创建签名

1. 使用 SHA-1 哈希函数和 RSA 对在为使用自定义策略的签名 URL 创建策略声明过程中创建的 JSON 策略声明进行哈希处理并签署。使用不再包含空格但尚未进行 Base64 编码的策略声明版本。

   对于哈希函数所需的私有密钥，请使用其公有密钥位于分配的活动可信密钥组中的私有密钥。

   > **Note**
   >
   > 您用于哈希及签署策略声明的方法取决于您的编程语言和平台。有关代码示例，请参阅为签名 URL 创建签名的代码示例。

2. 删除经过哈希处理并签署的字符串中的空格（包括制表符和换行符）。
3. 使用 MIME Base64 编码对字符串进行 Base64 编码。有关更多信息，请参阅 RFC 2045, MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies 中的 Section 6.8, Base64 Content-Transfer-Encoding。
4. 用有效的字符替换 URL 查询字符串中的无效字符。下表列出了无效和有效字符。

替换这些无效字符	使用这些有效字符
+	- (连字符)
=	_（下划线）
/	~（波浪字符）

5. 将结果值包含到 `Set-Cookie` 名称值对的 `CloudFront-Signature=` 标头中，并返回使用自定义策略设置签名 Cookie以添加 `Set-Cookie` 的 `CloudFront-Key-Pair-Id` 标头。

用于 Base64 编码和加密的 Linux 命令和 OpenSSL

可使用以下 Linux 命令行命令和 OpenSSL 哈希和签署策略声明，对签名进行 Base64 编码，将 URL 查询字符串中无效的字符替换为有效的字符。

有关 OpenSSL 的信息，请转到 https://www.openssl.org。

```
cat policy | tr -d "\n" | tr -d " \t\n\r" | openssl sha1 -sign private_key.pem |
 openssl base64 -A | tr -- '+=/' '-_~'
```

在上述命令中：

- `cat` 读取 `policy` 文件
- `tr -d "\n" | tr -d " \t\n\r"` 删除由 `cat` 添加的空格和换行符
- OpenSSL 使用 SHA-1 对文件进行哈希处理，并使用 RSA 和私有密钥文件 `private_key.pem` 对其进行签名
- OpenSSL 对经过哈希处理和签署的策略声明进行 Base64 编码
- `tr` 使用有效字符替换 URL 查询字符串参数中的无效字符

有关演示如何创建签名的更多代码示例，请参阅为签名 URL 创建签名的代码示例。

为签名 URL 创建签名的代码示例

本部分包括演示如何为签名 URL 创建签名的可下载应用程序示例。示例可以 Perl、PHP、C# 和 Java 语言提供。可以使用任意示例来创建签名 URL。Perl 脚本在 Linux 和 macOS 平台上运行。PHP 示例将在运行 PHP 的任何服务器上工作。C# 示例使用 .NET Framework。

有关 JavaScript (Node.js) 中的代码示例，请参阅 AWS 开发人员博客上的以 Node.js 创建 Amazon CloudFront 已签名 URL。

有关 Python 中的代码示例，请参阅 AWSSDK for Python (Boto3) API 参考中的为 Amazon CloudFront 生成签名 URL，以及 Boto3 GitHub 存储库中的此代码示例。

主题

- 使用 Perl 创建 URL 签名
- 使用 PHP 创建 URL 签名
- 使用 C# 和 .NET Framework 创建 URL 签名
- 使用 Java 创建 URL 签名

使用 Perl 创建 URL 签名

本部分包括一个用于 Linux/Mac 平台的 Perl 脚本，您可以使用它为私有内容创建签名。要创建签名，请使用指定 CloudFront URL、签署人私有密钥的路径、密钥 ID 以及 URL 的到期日期的命令行参数来运行该脚本。工具还可对签名 URL 进行解码。

> **Note**
>
> 创建 URL 签名只是使用签名 URL 提供私有内容过程的一部分。有关端到端流程的更多信息，请参阅使用签名 URL。

主题

- 用于创建签名 URL 的 Perl 脚本的源

用于创建签名 URL 的 Perl 脚本的源

以下 Perl 源代码可用于为 CloudFront 创建签名 URL。代码中的注释包括有关该工具的命令行开关和功能的信息。

```
#!/usr/bin/perl -w

# Copyright 2008 Amazon Technologies, Inc.  Licensed under the Apache License, Version
 2.0 (the "License");
# you may not use this file except in compliance with the License. You may obtain a
 copy of the License at:
#
# https://aws.amazon.com/apache2.0
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 KIND, either express or implied.
# See the License for the specific language governing permissions and limitations under
 the License.

=head1 cfsign.pl

cfsign.pl - A tool to generate and verify Amazon CloudFront signed URLs

=head1 SYNOPSIS

This script uses an existing RSA key pair to sign and verify Amazon CloudFront signed
 URLs

View the script source for details as to which CPAN packages are required beforehand.

For help, try:
```

```
cfsign.pl --help

URL signing examples:

cfsign.pl --action encode --url https://images.my-website.com/gallery1.zip --policy
 sample_policy.json --private-key privkey.pem --key-pair-id mykey

cfsign.pl --action encode --url https://images.my-website.com/gallery1.zip --expires
 1257439868 --private-key privkey.pem --key-pair-id mykey

URL decode example:

cfsign.pl --action decode --url "http//mydist.cloudfront.net/?Signature=AGO-
PgxkYo99MkJFHvjfGXjG1QDEXeaDb4Qtzmy85wqyJjK7eKojQWa4BCRcow__&Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc29
Pair-Id=mykey"


To generate an RSA key pair, you can use openssl and the following commands:

# Generate a 2048 bit key pair
openssl genrsa -out private-key.pem 2048
openssl rsa -in private-key.pem -pubout -out public-key.pem


=head1 OPTIONS

=over 8

=item B<--help>

Print a help message and exits.

=item B<--action> [action]

The action to execute.  action can be one of:

  encode - Generate a signed URL (using a canned policy or a user policy)
  decode - Decode a signed URL

=item B<--url>

The URL to en/decode

=item B<--stream>
```

```
The stream to en/decode

=item B<--private-key>

The path to your private key.

=item B<--key-pair-id>

The key pair identifier.

=item B<--policy>

The CloudFront policy document.

=item B<--expires>

The Unix epoch time when the URL is to expire. If both this option and
the --policy option are specified, --policy will be used. Otherwise, this
option alone will use a canned policy.

=back

=cut

use strict;
use warnings;

# you might need to use CPAN to get these modules.
# run perl -MCPAN -e "install <module>" to get them.
# The openssl command line will also need to be in your $PATH.
use File::Temp qw/tempfile/;
use File::Slurp;
use Getopt::Long;
use IPC::Open2;
use MIME::Base64 qw(encode_base64 decode_base64);
use Pod::Usage;
use URI;

my $CANNED_POLICY
    = '{"Statement":[{"Resource":"<RESOURCE>","Condition":{"DateLessThan":
{"AWS:EpochTime":<EXPIRES>}}}]}';

my $POLICY_PARAM      = "Policy";
```

```
my $EXPIRES_PARAM     = "Expires";
my $SIGNATURE_PARAM   = "Signature";
my $KEY_PAIR_ID_PARAM = "Key-Pair-Id";

my $verbose = 0;
my $policy_filename = "";
my $expires_epoch = 0;
my $action = "";
my $help = 0;
my $key_pair_id = "";
my $url = "";
my $stream = "";
my $private_key_filename = "";

my $result = GetOptions("action=s"      => \$action,
                        "policy=s"      => \$policy_filename,
                        "expires=i"     => \$expires_epoch,
                        "private-key=s" => \$private_key_filename,
                        "key-pair-id=s" => \$key_pair_id,
                        "verbose"       => \$verbose,
                        "help"          => \$help,
                        "url=s"         => \$url,
                        "stream=s"      => \$stream,
                    );

if ($help or !$result) {
    pod2usage(1);
    exit;
}

if ($url eq "" and $stream eq "") {
    print STDERR "Must include a stream or a URL to encode or decode with the --stream
 or --url option\n";
    exit;
}

if ($url ne "" and $stream ne "") {
    print STDERR "Only one of --url and --stream may be specified\n";
    exit;
}

if ($url ne "" and !is_url_valid($url)) {
    exit;
}
```

```
if ($stream ne "") {
    exit unless is_stream_valid($stream);

    # The signing mechanism is identical, so from here on just pretend we're
    # dealing with a URL
    $url = $stream;
}

if ($action eq "encode") {
    # The encode action will generate a private content URL given a base URL,
    # a policy file (or an expires timestamp) and a key pair id parameter
    my $private_key;
    my $public_key;
    my $public_key_file;

    my $policy;
    if ($policy_filename eq "") {
        if ($expires_epoch == 0) {
            print STDERR "Must include policy filename with --policy argument or an
 expires" .
                         "time using --expires\n";
        }

        $policy = $CANNED_POLICY;
        $policy =~ s/<EXPIRES>/$expires_epoch/g;
        $policy =~ s/<RESOURCE>/$url/g;
    } else {
        if (! -e $policy_filename) {
            print STDERR "Policy file $policy_filename does not exist\n";
            exit;
        }
        $expires_epoch = 0; # ignore if set
        $policy = read_file($policy_filename);
    }

    if ($private_key_filename eq "") {
        print STDERR "You must specific the path to your private key file with --
private-key\n";
        exit;
    }

    if (! -e $private_key_filename) {
        print STDERR "Private key file $private_key_filename does not exist\n";
```

```
        exit;
    }

    if ($key_pair_id eq "") {
        print STDERR "You must specify a key pair id with --key-pair-id\n";
        exit;
    }

    my $encoded_policy = url_safe_base64_encode($policy);
    my $signature = rsa_sha1_sign($policy, $private_key_filename);
    my $encoded_signature = url_safe_base64_encode($signature);

    my $generated_url = create_url($url, $encoded_policy, $encoded_signature,
 $key_pair_id, $expires_epoch);


    if ($stream ne "") {
        print "Encoded stream (for use within a swf):\n" . $generated_url . "\n";
        print "Encoded and escaped stream (for use on a webpage):\n" .
 escape_url_for_webpage($generated_url) . "\n";
    } else {
        print "Encoded URL:\n" . $generated_url . "\n";
    }
} elsif ($action eq "decode") {
    my $decoded = decode_url($url);
    if (!$decoded) {
        print STDERR "Improperly formed URL\n";
        exit;
    }

    print_decoded_url($decoded);
} else {
    # No action specified, print help.  But only if this is run as a program (caller
 will be empty)
    pod2usage(1) unless caller();
}

# Decode a private content URL into its component parts
sub decode_url {
    my $url = shift;

    if ($url =~ /(.*)\?(.*)/) {
        my $base_url = $1;
        my $params = $2;
```

```
        my @unparsed_params = split(/&/, $params);
        my %params = ();
        foreach my $param (@unparsed_params) {
            my ($key, $val) = split(/=/, $param);
            $params{$key} = $val;
        }

        my $encoded_signature = "";
        if (exists $params{$SIGNATURE_PARAM}) {
            $encoded_signature = $params{"Signature"};
        } else {
            print STDERR "Missing Signature URL parameter\n";
            return 0;
        }

        my $encoded_policy = "";
        if (exists $params{$POLICY_PARAM}) {
            $encoded_policy = $params{$POLICY_PARAM};
        } else {
            if (!exists $params{$EXPIRES_PARAM}) {
                print STDERR "Either the Policy or Expires URL parameter needs to be
specified\n";
                return 0;
            }

            my $expires = $params{$EXPIRES_PARAM};

            my $policy = $CANNED_POLICY;
            $policy =~ s/<EXPIRES>/$expires/g;

            my $url_without_cf_params = $url;
            $url_without_cf_params =~ s/$SIGNATURE_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$POLICY_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$EXPIRES_PARAM=[^&]*&?//g;
            $url_without_cf_params =~ s/$KEY_PAIR_ID_PARAM=[^&]*&?//g;

            if ($url_without_cf_params =~ /(.*)\?$/) {
                $url_without_cf_params = $1;
            }

            $policy =~ s/<RESOURCE>/$url_without_cf_params/g;

            $encoded_policy = url_safe_base64_encode($policy);
```

```
        }

        my $key = "";
        if (exists $params{$KEY_PAIR_ID_PARAM}) {
            $key = $params{$KEY_PAIR_ID_PARAM};
        } else {
            print STDERR "Missing $KEY_PAIR_ID_PARAM parameter\n";
            return 0;
        }

        my $policy = url_safe_base64_decode($encoded_policy);

        my %ret = ();
        $ret{"base_url"} = $base_url;
        $ret{"policy"} = $policy;
        $ret{"key"} = $key;

        return \%ret;
    } else {
        return 0;
    }
}

# Print a decoded URL out
sub print_decoded_url {
    my $decoded = shift;

    print "Base URL: \n" . $decoded->{"base_url"} . "\n";
    print "Policy: \n" . $decoded->{"policy"} . "\n";
    print "Key: \n" . $decoded->{"key"} . "\n";
}

# Encode a string with base 64 encoding and replace some invalid URL characters
sub url_safe_base64_encode {
    my ($value) = @_;

    my $result = encode_base64($value);
    $result =~ tr|+=/|-_~|;

    return $result;
}

# Decode a string with base 64 encoding.  URL-decode the string first
# followed by reversing any special character ("+=/") translation.
```

```
sub url_safe_base64_decode {
    my ($value) = @_;

    $value =~ s/%([0-9A-Fa-f]{2})/chr(hex($1))/eg;
    $value =~ tr|-_~|+=/|;

    my $result = decode_base64($value);

    return $result;
}

# Create a private content URL
sub create_url {
    my ($path, $policy, $signature, $key_pair_id, $expires) = @_;

    my $result;
    my $separator = $path =~ /\?/ ? '&' : '?';
    if ($expires) {
        $result = "$path$separator$EXPIRES_PARAM=$expires&$SIGNATURE_PARAM=$signature&
$KEY_PAIR_ID_PARAM=$key_pair_id";
    } else {
        $result = "$path$separator$POLICY_PARAM=$policy&$SIGNATURE_PARAM=$signature&
$KEY_PAIR_ID_PARAM=$key_pair_id";
    }
    $result =~ s/\n//g;

    return $result;
}

# Sign a document with given private key file.
# The first argument is the document to sign
# The second argument is the name of the private key file
sub rsa_sha1_sign {
    my ($to_sign, $pvkFile) = @_;
    print "openssl sha1 -sign $pvkFile $to_sign\n";

    return write_to_program($pvkFile, $to_sign);
}

# Helper function to write data to a program
sub write_to_program {
my ($keyfile, $data) = @_;
unlink "temp_policy.dat" if (-e "temp_policy.dat");
unlink "temp_sign.dat" if (-e "temp_sign.dat");
```

```
write_file("temp_policy.dat", $data);

system("openssl dgst -sha1 -sign \"$keyfile\" -out temp_sign.dat temp_policy.dat");

my $output = read_file("temp_sign.dat");

    return $output;
}

# Read a file into a string and return the string
sub read_file {
    my ($file) = @_;

    open(INFILE, "<$file") or die("Failed to open $file: $!");
    my $str = join('', <INFILE>);
    close INFILE;

    return $str;
}

sub is_url_valid {
    my ($url) = @_;

    # HTTP distributions start with http[s]:// and are the correct thing to sign
    if ($url =~ /^https?:\/\//) {
        return 1;
    } else {
        print STDERR "CloudFront requires absolute URLs for HTTP distributions\n";
        return 0;
    }
}

sub is_stream_valid {
    my ($stream) = @_;

    if ($stream =~ /^rtmp:\/\// or $stream =~ /^\/?cfx\/st/) {
        print STDERR "Streaming distributions require that only the stream name is
 signed.\n";
        print STDERR "The stream name is everything after, but not including, cfx/st/
\n";
        return 0;
    } else {
        return 1;
```

```
    }
}

# flash requires that the query parameters in the stream name are url
# encoded when passed in through javascript, etc.  This sub handles the minimal
# required url encoding.
sub escape_url_for_webpage {
    my ($url) = @_;

    $url =~ s/\?/%3F/g;
    $url =~ s/=/%3D/g;
    $url =~ s/&/%26/g;

    return $url;
}

1;
```

使用 PHP 创建 URL 签名

运行 PHP 的任何 Web 服务器可使用此 PHP 代码示例，来为私有的 CloudFront 分配创建策略声明和签名。该完整示例使用签名 URL 链接创建一个正常运作的网页，这些链接使用 CloudFront 流播放视频流。您可以在下面下载完整示例：https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/samples/demo-php.zip。

还可以通过使用 AWS SDK for PHP 中的 `UrlSigner` 类来创建签名 URL。有关更多信息，请参阅 AWS SDK for PHPAPI 参考中的 Class UrlSigner。

> **Note**
>
> 创建 URL 签名只是使用签名 URL 提供私有内容过程的一部分。有关整个过程的更多信息，请参阅使用签名 URL。

主题

示例：RSA SHA-1 签名

在以下代码示例中，函数 rsa_sha1_sign 哈希并签署策略声明。所需的参数是策略语句和私有密钥，该私有密钥对应于分配的可信密钥组中的公有密钥。接下来，url_safe_base64_encode 函数创建签名 URL 安全版本。

```
function rsa_sha1_sign($policy, $private_key_filename) {
    $signature = "";

    // load the private key
    $fp = fopen($private_key_filename, "r");
    $priv_key = fread($fp, 8192);
    fclose($fp);
    $pkeyid = openssl_get_privatekey($priv_key);

    // compute signature
    openssl_sign($policy, $signature, $pkeyid);

    // free the key from memory
    openssl_free_key($pkeyid);

    return $signature;
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with
    // the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}
```

示例：创建标准策略

以下代码示例构建了签名的标准策略声明。更多有关标准策略的信息，请参阅 使用标准策略创建签名 URL。

> **Note**
>
> $expires 变量是一个日期/时间戳，必须为整数，而不是字符串。

```
function get_canned_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it,
    // since it contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condition":
{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';

    // sign the canned policy
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, null, $encoded_signature,
 $key_pair_id, $expires);
    // url-encode the query string characters to work around a flash player bug
    return encode_query_params($stream_name);
}
```

示例：创建自定义策略

以下代码示例构建了签名的自定义 策略声明。更多有关自定义策略的信息，请参阅 使用自定义策略创建签名 URL。

```
function get_custom_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $policy) {
    // sign the policy
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a url
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_signature,
 $key_pair_id, null);
    // url-encode the query string characters to work around a flash player bug
    return encode_query_params($stream_name);
}
```

完整代码示例

以下代码示例完整演示了使用 PHP 创建 CloudFront 签名 URL。您可以在下面下载此完整示例：https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/samples/demo-php.zip。

在下面的示例中，您可以修改 $policy Condition 元素以允许 IPv4 和 IPv6 地址范围。有关示例，请参阅《Amazon Simple Storage Service 用户指南》中的在 IAM 策略中使用 IPv6 地址。

```
<?php

function rsa_sha1_sign($policy, $private_key_filename) {
    $signature = "";

    // load the private key
    $fp = fopen($private_key_filename, "r");
    $priv_key = fread($fp, 8192);
    fclose($fp);
    $pkeyid = openssl_get_privatekey($priv_key);

    // compute signature
    openssl_sign($policy, $signature, $pkeyid);

    // free the key from memory
    openssl_free_key($pkeyid);

    return $signature;
}

function url_safe_base64_encode($value) {
    $encoded = base64_encode($value);
    // replace unsafe characters +, = and / with the safe characters -, _ and ~
    return str_replace(
        array('+', '=', '/'),
        array('-', '_', '~'),
        $encoded);
}

function create_stream_name($stream, $policy, $signature, $key_pair_id, $expires) {
    $result = $stream;
    // if the stream already contains query parameters, attach the new query parameters
 to the end
    // otherwise, add the query parameters
```

```
    $separator = strpos($stream, '?') == FALSE ? '?' : '&';
    // the presence of an expires time means we're using a canned policy
    if($expires) {
        $result .= $path . $separator . "Expires=" . $expires . "&Signature=" .
 $signature . "&Key-Pair-Id=" . $key_pair_id;
    }
    // not using a canned policy, include the policy itself in the stream name
    else {
        $result .= $path . $separator . "Policy=" . $policy . "&Signature=" .
 $signature . "&Key-Pair-Id=" . $key_pair_id;
    }

    // new lines would break us, so remove them
    return str_replace('\n', '', $result);
}

function encode_query_params($stream_name) {
    // Adobe Flash Player has trouble with query parameters being passed into it,
    // so replace the bad characters with their URL-encoded forms
    return str_replace(
        array('?', '=', '&'),
        array('%3F', '%3D', '%26'),
        $stream_name);
}

function get_canned_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $expires) {
    // this policy is well known by CloudFront, but you still need to sign it, since it
 contains your parameters
    $canned_policy = '{"Statement":[{"Resource":"' . $video_path . '","Condition":
{"DateLessThan":{"AWS:EpochTime":'. $expires . '}}}]}';
    // the policy contains characters that cannot be part of a URL, so we base64 encode
 it
    $encoded_policy = url_safe_base64_encode($canned_policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($canned_policy, $private_key_filename);
    // make the signature safe to be included in a URL
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, null, $encoded_signature,
 $key_pair_id, $expires);
    // URL-encode the query string characters to support Flash Player
    return encode_query_params($stream_name);
```

```
}

function get_custom_policy_stream_name($video_path, $private_key_filename,
 $key_pair_id, $policy) {
    // the policy contains characters that cannot be part of a URL, so we base64 encode
 it
    $encoded_policy = url_safe_base64_encode($policy);
    // sign the original policy, not the encoded version
    $signature = rsa_sha1_sign($policy, $private_key_filename);
    // make the signature safe to be included in a URL
    $encoded_signature = url_safe_base64_encode($signature);

    // combine the above into a stream name
    $stream_name = create_stream_name($video_path, $encoded_policy, $encoded_signature,
 $key_pair_id, null);
    // URL-encode the query string characters to support Flash Player
    return encode_query_params($stream_name);
}


// Path to your private key.  Be very careful that this file is not accessible
// from the web!

$private_key_filename = '/home/test/secure/example-priv-key.pem';
$key_pair_id = 'K2JCJMDEHXQW5F';

$video_path = 'example.mp4';

$expires = time() + 300; // 5 min from now
$canned_policy_stream_name = get_canned_policy_stream_name($video_path,
 $private_key_filename, $key_pair_id, $expires);

$client_ip = $_SERVER['REMOTE_ADDR'];
$policy =
'{'.
    '"Statement":['.
        '{'.
            '"Resource":"'. $video_path . '",'.
            '"Condition":{'.
                '"IpAddress":{"AWS:SourceIp":"' . $client_ip . '/32"},'.
                '"DateLessThan":{"AWS:EpochTime":' . $expires . '}'.
            '}'.
        '}'.
    ']' .
```

```
    '}';
$custom_policy_stream_name = get_custom_policy_stream_name($video_path,
 $private_key_filename, $key_pair_id, $policy);

?>

<html>

<head>
    <title>CloudFront</title>
<script type='text/javascript' src='https://example.cloudfront.net/player/
swfobject.js'></script>
</head>

<body>
    <h1>Amazon CloudFront</h1>
    <h2>Canned Policy</h2>
    <h3>Expires at <?= gmdate('Y-m-d H:i:s T', $expires) ?></h3>
    <br />

    <div id='canned'>The canned policy video will be here</div>

    <h2>Custom Policy</h2>
    <h3>Expires at <?= gmdate('Y-m-d H:i:s T', $expires) ?> only viewable by IP <?=
 $client_ip ?></h3>
    <div id='custom'>The custom policy video will be here</div>

    <!-- ************** Have to update the player.swf path to a real JWPlayer instance.
    The fake one means that external people cannot watch the video right now -->
    <script type='text/javascript'>
  var so_canned = new SWFObject('https://files.example.com/
player.swf','mpl','640','360','9');
  so_canned.addParam('allowfullscreen','true');
  so_canned.addParam('allowscriptaccess','always');
  so_canned.addParam('wmode','opaque');
  so_canned.addVariable('file','<?= $canned_policy_stream_name ?>');
  so_canned.addVariable('streamer','rtmp://example.cloudfront.net/cfx/st');
        so_canned.write('canned');

  var so_custom = new SWFObject('https://files.example.com/
player.swf','mpl','640','360','9');
  so_custom.addParam('allowfullscreen','true');
  so_custom.addParam('allowscriptaccess','always');
  so_custom.addParam('wmode','opaque');
```

```
  so_custom.addVariable('file','<?= $custom_policy_stream_name ?>');
  so_custom.addVariable('streamer','rtmp://example.cloudfront.net/cfx/st');
  so_custom.write('custom');
    </script>
</body>

</html>
```

另请参阅：

- 使用 Perl 创建 URL 签名
- 使用 C# 和 .NET Framework 创建 URL 签名
- 使用 Java 创建 URL 签名

使用 C# 和 .NET Framework 创建 URL 签名

本节中的 C# 示例实现了一个示例应用程序，该程序演示了如何使用标准和自定义策略声明为 CloudFront 私有分配创建签名。此示例包括基于 AWS SDK for .NET 的实用工具函数，这些函数在 .NET 应用程序中非常有用。

还可以通过使用 AWS SDK for .NET 创建签名 URL 和签名 Cookie。在 AWS SDK for .NET API 参考中，请参阅以下主题：

- 签名 URL – AmazonCloudFrontUrlSigner
- 签名 Cookie – AmazonCloudFrontCookieSigner

要下载代码，请转到 C# 签名代码。

> **Note**
>
> 创建 URL 签名只是使用签名 URL 提供私有内容过程的一部分。有关整个过程的更多信息，请参阅使用签名 URL。有关使用签名 Cookie 的更多信息，请参阅使用签名 Cookie。

在 .NET 框架中使用 RSA 密钥

要在 .NET 框架中使用 RSA 密钥，您必须将 AWS 提供的 .pem 文件转换为 .NET 框架使用的 XML 格式。

转换之后，RSA 私有密钥文件是以下格式：

Example ：XML .NET 框架格式的 RSA 私有密钥

```
<RSAKeyValue>
  <Modulus>
    wO5IvYCP5UcoCKDo1dcspoMehWBZcyfs9QEzGi6Oe5y+ewGr1oW+vB2GPB
    ANBiVPcUHTFWhwaIBd3oglmF0lGQljP/jOfmXHUK2kUUnLnJp+oOBL2NiuFtqcW6h/L5lIpD8Yq+NRHg
    Ty4zDsyr2880MvXv88yEFURCkqEXAMPLE=
  </Modulus>
  <Exponent>AQAB</Exponent>
  <P>
    5bmKDaTz
    npENGVqz4Cea8XPH+sxt+2VaAwYnsarVUoSBeVt8WLloVuZGG9IZYmH5KteXEu7fZveYd9UEXAMPLE==
  </P>
  <Q>
    1v9l/WN1a1N3rOK4VGoCokx7kR2SyTMSbZgF9IWJNOugR/WZw7HTnjipO3c9dy1Ms9pUKwUF4
    6d7049EXAMPLE==
  </Q>
  <DP>
    RgrSKuLWXMyBH+/l1Dx/I4tXuAJIrlPyo+VmiOc7b5NzHptkSHEPfR9s1
    OK0VqjknclqCJ3Ig86OMEtEXAMPLE==
  </DP>
  <DQ>
    pjPjvSFw+RoaTu0pgCA/jwW/FGyfN6iim1RFbkT4
    z49DZb2IM885f3vf35eLTaEYRYUHQgZtChNEV0TEXAMPLE==
  </DQ>
  <InverseQ>
    nkvOJTg5QtGNgWb9i
    cVtzrL/1pFEOHbJXwEJdU99N+7sMK+1066DL/HSBUCD63qD4USpnf0myc24in0EXAMPLE==</InverseQ>
  <D>
      Bc7mp7XYHynuPZxChjWNJZIq+A73gm0ASDv6At7F8Vi9r0xUlQe/v0AQS3ycN8QlyR4XMbzMLYk
      3yjxFDXo4ZKQtOGzLGteCU2srANiLv26/imXA8FVidZftTAtLviWQZBVPTeYIA69ATUYPEq0a5u5wjGy
      UOij9OWyuEXAMPLE=
   </D>
</RSAKeyValue>
```

C# 中的标准策略签名方法

以下 C# 代码通过执行以下步骤创建使用标准策略的签名 URL：

- 创建策略声明。

- 使用 SHA1 对策略声明进行哈希处理，并使用 RSA 和其相应公有密钥位于可信密钥组中的私有密钥对结果进行签名。
- 对经过哈希及签署的策略声明进行 Base64 编码，并替换使字符串能够安全地用作 URL 请求参数的特殊字符。
- 联接值。

有关完整实现，请参阅 C# 签名代码中的示例。

> **Note**
>
> 当您将公有密钥上传到 CloudFront 时，会返回 `keyId`。有关更多信息，请参阅
>
> 6
>
> &Key-Pair-Id。

Example ：C# 中的标准策略签名方法

```
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '_')
        .Replace('/', '~');
}

public static string CreateCannedPrivateURL(string urlString,
    string durationUnits, string durationNumber, string pathToPolicyStmnt,
    string pathToPrivateKey, string keyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-pathToPolicyStmnt,
    // 5-pathToPrivateKey, 6-keyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);

    // Create the policy statement.
    string strPolicy = CreatePolicyStatement(pathToPolicyStmnt,
        urlString,
        DateTime.Now,
        DateTime.Now.Add(timeSpanInterval),
```

```
        "0.0.0.0/0");
    if ("Error!" == strPolicy) return "Invalid time frame." +
        "Start time cannot be greater than end time.";

    // Copy the expiration time defined by policy statement.
    string strExpiration = CopyExpirationTimeFromPolicy(strPolicy);

    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.
    using (SHA1CryptoServiceProvider
        cryptoSHA1 = new SHA1CryptoServiceProvider())
    {
        bufferPolicy = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();
        XmlDocument xmlPrivateKey = new XmlDocument();

        // Load your private key, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
        // Several tools are available.
        xmlPrivateKey.Load(pathToPrivateKey);

        // Format the RSACryptoServiceProvider providerRSA and
        // create the signature.
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
        RSAPKCS1SignatureFormatter rsaFormatter =
            new RSAPKCS1SignatureFormatter(providerRSA);
        rsaFormatter.SetHashAlgorithm("SHA1");
        byte[] signedPolicyHash = rsaFormatter.CreateSignature(bufferPolicy);

        // Convert the signed policy to URL-safe base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~
        string strSignedPolicy = ToUrlSafeBase64String(signedPolicyHash);

        // Concatenate the URL, the timestamp, the signature,
        // and the key pair ID to form the signed URL.
        return urlString +
            "?Expires=" +
            strExpiration +
            "&Signature=" +
            strSignedPolicy +
```

```
            "&Key-Pair-Id=" +
            keyId;
    }
}
```

C# 中自定义策略签名方法

以下 C# 代码通过执行以下步骤创建使用自定义策略的签名 URL：

1. 创建策略声明。
2. 对策略声明进行 Base64 编码，并替换使字符串能够安全地用作 URL 请求参数的特殊字符。
3. 使用 SHA1 对策略声明进行哈希处理，并使用 RSA 和其相应公有密钥位于可信密钥组中的私有密钥对结果进行加密。
4. 对经过哈希的策略声明进行 Base64 编码，并替换使字符串能够安全地用作 URL 请求参数的特殊字符。
5. 联接值。

有关完整实现，请参阅 C# 签名代码中的示例。

> **Note**
>
> 当您将公有密钥上传到 CloudFront 时，会返回 `keyId`。有关更多信息，请参阅 &Key-Pair-Id。

Example ：C# 中的自定义策略签名方法

```
public static string ToUrlSafeBase64String(byte[] bytes)
{
    return System.Convert.ToBase64String(bytes)
        .Replace('+', '-')
        .Replace('=', '_')
        .Replace('/', '~');
}

public static string CreateCustomPrivateURL(string urlString,
    string durationUnits, string durationNumber, string startIntervalFromNow,
    string ipaddress, string pathToPolicyStmnt, string pathToPrivateKey,
```

```
    string keyId)
{
    // args[] 0-thisMethod, 1-resourceUrl, 2-seconds-minutes-hours-days
    // to expiration, 3-numberOfPreviousUnits, 4-starttimeFromNow,
    // 5-ip_address, 6-pathToPolicyStmt, 7-pathToPrivateKey, 8-keyId

    TimeSpan timeSpanInterval = GetDuration(durationUnits, durationNumber);
    TimeSpan timeSpanToStart = GetDurationByUnits(durationUnits,
        startIntervalFromNow);
    if (null == timeSpanToStart)
        return "Invalid duration units." +
            "Valid options: seconds, minutes, hours, or days";

    string strPolicy = CreatePolicyStatement(
        pathToPolicyStmnt, urlString, DateTime.Now.Add(timeSpanToStart),
        DateTime.Now.Add(timeSpanInterval), ipaddress);

    // Read the policy into a byte buffer.
    byte[] bufferPolicy = Encoding.ASCII.GetBytes(strPolicy);

    // Convert the policy statement to URL-safe base64 encoding and
    // replace unsafe characters + = / with the safe characters - _ ~

    string urlSafePolicy = ToUrlSafeBase64String(bufferPolicy);

    // Initialize the SHA1CryptoServiceProvider object and hash the policy data.
    byte[] bufferPolicyHash;
    using (SHA1CryptoServiceProvider cryptoSHA1 =
        new SHA1CryptoServiceProvider())
    {
        bufferPolicyHash = cryptoSHA1.ComputeHash(bufferPolicy);

        // Initialize the RSACryptoServiceProvider object.
        RSACryptoServiceProvider providerRSA = new RSACryptoServiceProvider();
        XmlDocument xmlPrivateKey = new XmlDocument();

        // Load your private key, which you created by converting your
        // .pem file to the XML format that the .NET framework uses.
        // Several tools are available.
        xmlPrivateKey.Load(pathToPrivateKey);

        // Format the RSACryptoServiceProvider providerRSA
        // and create the signature.
        providerRSA.FromXmlString(xmlPrivateKey.InnerXml);
```

```
        RSAPKCS1SignatureFormatter RSAFormatter =
            new RSAPKCS1SignatureFormatter(providerRSA);
        RSAFormatter.SetHashAlgorithm("SHA1");
        byte[] signedHash = RSAFormatter.CreateSignature(bufferPolicyHash);

        // Convert the signed policy to URL-safe base64 encoding and
        // replace unsafe characters + = / with the safe characters - _ ~
        string strSignedPolicy = ToUrlSafeBase64String(signedHash);

        return urlString +
            "?Policy=" +
            urlSafePolicy +
            "&Signature=" +
            strSignedPolicy +
            "&Key-Pair-Id=" +
            keyId;
    }
}
```

用于签名生成的实用工具方法

以下方法从文件和签名生成解析时间间隔中获得策略声明。

Example ：用于签名生成的实用工具方法

```
public static string CreatePolicyStatement(string policyStmnt,
   string resourceUrl,
   DateTime startTime,
   DateTime endTime,
   string ipAddress)

{
   // Create the policy statement.
   FileStream streamPolicy = new FileStream(policyStmnt, FileMode.Open,
 FileAccess.Read);
   using (StreamReader reader = new StreamReader(streamPolicy))
   {
      string strPolicy = reader.ReadToEnd();

      TimeSpan startTimeSpanFromNow = (startTime - DateTime.Now);
      TimeSpan endTimeSpanFromNow = (endTime - DateTime.Now);
      TimeSpan intervalStart =
         (DateTime.UtcNow.Add(startTimeSpanFromNow)) -
         new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);
```

```
      TimeSpan intervalEnd =
         (DateTime.UtcNow.Add(endTimeSpanFromNow)) -
         new DateTime(1970, 1, 1, 0, 0, 0, DateTimeKind.Utc);

      int startTimestamp = (int)intervalStart.TotalSeconds; // START_TIME
      int endTimestamp = (int)intervalEnd.TotalSeconds;  // END_TIME

      if (startTimestamp > endTimestamp)
         return "Error!";

      // Replace variables in the policy statement.
      strPolicy = strPolicy.Replace("RESOURCE", resourceUrl);
      strPolicy = strPolicy.Replace("START_TIME", startTimestamp.ToString());
      strPolicy = strPolicy.Replace("END_TIME", endTimestamp.ToString());
      strPolicy = strPolicy.Replace("IP_ADDRESS", ipAddress);
      strPolicy = strPolicy.Replace("EXPIRES", endTimestamp.ToString());
      return strPolicy;
   }
}

public static TimeSpan GetDuration(string units, string numUnits)
{
   TimeSpan timeSpanInterval = new TimeSpan();
   switch (units)
   {
      case "seconds":
         timeSpanInterval = new TimeSpan(0, 0, 0, int.Parse(numUnits));
         break;
      case "minutes":
         timeSpanInterval = new TimeSpan(0, 0, int.Parse(numUnits), 0);
         break;
      case "hours":
         timeSpanInterval = new TimeSpan(0, int.Parse(numUnits), 0 ,0);
         break;
      case "days":
         timeSpanInterval = new TimeSpan(int.Parse(numUnits),0 ,0 ,0);
         break;
      default:
         Console.WriteLine("Invalid time units;" +
            "use seconds, minutes, hours, or days");
         break;
   }
   return timeSpanInterval;
}
```

```
private static TimeSpan GetDurationByUnits(string durationUnits,
   string startIntervalFromNow)
{
   switch (durationUnits)
   {
      case "seconds":
         return new TimeSpan(0, 0, int.Parse(startIntervalFromNow));
      case "minutes":
         return new TimeSpan(0, int.Parse(startIntervalFromNow), 0);
      case "hours":
         return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0);
      case "days":
         return new TimeSpan(int.Parse(startIntervalFromNow), 0, 0, 0);
      default:
         return new TimeSpan(0, 0, 0, 0);
   }
}

public static string CopyExpirationTimeFromPolicy(string policyStatement)
{
   int startExpiration = policyStatement.IndexOf("EpochTime");
   string strExpirationRough = policyStatement.Substring(startExpiration +
      "EpochTime".Length);
   char[] digits = { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };

   List<char> listDigits = new List<char>(digits);
   StringBuilder buildExpiration = new StringBuilder(20);

   foreach (char c in strExpirationRough)
   {
      if (listDigits.Contains(c))
         buildExpiration.Append(c);
   }
   return buildExpiration.ToString();
}
```

另请参阅

- 使用 Perl 创建 URL 签名
- 使用 PHP 创建 URL 签名
- 使用 Java 创建 URL 签名

使用 Java 创建 URL 签名

除了以下代码示例外，您还可以使用 AWS SDK for Java（版本 1）中的 `CloudFrontUrlSigner` 实用程序类来创建 CloudFront 签名 URL。

有关更多示例，请参阅《AWS SDK 代码示例代码库》中的使用 AWS SDK 创建签名 URL 和 Cookie。

> **Note**
>
> 创建签名 URL 只是使用使用 CloudFront 提供私有内容过程的一部分。有关整个过程的更多信息，请参阅使用签名 URL。

以下示例演示如何创建 CloudFront 签名 URL。

Example Java 策略和签名加密方法

```
package org.example;

import java.time.Instant;
import java.time.temporal.ChronoUnit;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class Main {

    public static void main(String[] args) throws Exception {
        CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        String resourceUrl = "https://a1b2c3d4e5f6g7.cloudfront.net";
        String keyPairId = "K1UA3WV15I7JSD";
        CannedSignerRequest cannedRequest = CannedSignerRequest.builder()
                .resourceUrl(resourceUrl)
                .privateKey(new java.io.File("/path/to/private_key.pem").toPath())
                .keyPairId(keyPairId)
                .expirationDate(expirationDate)
                .build();
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest);
        String url = signedUrl.url();
        System.out.println(url);
```

```
    }
}
```

另请参阅：

- 使用 Perl 创建 URL 签名
- 使用 PHP 创建 URL 签名
- 使用 C# 和 .NET Framework 创建 URL 签名

限制对AWS源的访问

您可以通过可提供以下好处的方式配置 CloudFront 和某些 AWS 源：

- 限制对 AWS 源的访问，使其无法供公开访问
- 确保查看者（用户）只能通过指定的 CloudFront 分配访问 AWS 源中的内容 — 防止他们直接或通过意外的 CloudFront 分配访问桶中的内容

为此，请将 CloudFront 配置为向 AWS 源发送经过身份验证的请求，并将 AWS 源配置为仅允许访问来自 CloudFront 的经过身份验证的请求。有关更多信息，请参阅以下主题，了解 AWS 源的兼容类型。

主题

- 限制对 AWS Elemental MediaPackage v2 源的访问
- 限制对AWS Elemental MediaStore源的访问
- 限制对 AWS Lambda 函数 URL 源的访问
- 限制对 Amazon Simple Storage Service 源的访问

限制对 AWS Elemental MediaPackage v2 源的访问

CloudFront 提供源访问控制（OAC），用于限制对 MediaPackage v2 源的访问。

> **Note**
>
> CloudFront OAC 仅支持 MediaPackage v2。不支持 MediaPackage v1。

主题

创建新的 OAC

完成以下主题中描述的步骤，在 CloudFront 中设置新的 OAC。

主题

先决条件

在创建和设置 OAC 之前，您必须拥有带 MediaPackage v2 源的 CloudFront 分配。有关更多信息，请参阅使用 MediaStore 容器或 MediaPackage 通道。

向 OAC 授予访问 MediaPackage v2 源的权限

在创建 OAC 或在 CloudFront 分配中设置它之前，请确保 OAC 有权访问 MediaPackage v2 源。请在创建 CloudFront 分配后，但在分配配置中将 OAC 添加到 MediaPackage v2 源之前，执行此操作。

要授予 OAC 访问 MediaPackage v2 源的权限，请使用 IAM 策略，以允许 CloudFront 服务主体（`cloudfront.amazonaws.com`）访问源。使用策略中的 `Condition` 元素，仅 在该请求代表包含 MediaPackage v2 源的 CloudFront 分配时，才允许 CloudFront 访问 MediaPackage v2 源。

Example ：允许对 CloudFront 分配进行只读访问的 IAM 策略

以下策略允许 CloudFront 分配（*`E1PDK09ESKHJWT`*）访问 MediaPackage v2 源。该源是为 `Resource` 元素指定的 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipal",
            "Effect": "Allow",
            "Principal": {"Service": "cloudfront.amazonaws.com"},
```

```
            "Action": "mediapackagev2:GetObject",
            "Resource": "arn:aws:mediapackagev2:us-
east-1:123456789012:channelGroup/channel-group-name/channel/channel-name/
originEndpoint/origin_endpoint_name",
            "Condition": {
                "StringEquals": {"AWS:SourceArn":
 "arn:aws:cloudfront::123456789012:distribution/E1PDK09ESKHJWT"}
            }
        }
    ]
}
```

> **Note**
>
> 如果您创建的分配无权访问您的 MediaPackage v2 源，则可以从 CloudFront 控制台中选择复制策略，然后选择更新端点权限。然后，您可以将复制的权限附加到端点。有关更多信息，请参阅《AWS Elemental MediaPackage 用户指南》中的端点策略字段。

创建 OAC

要创建 OAC，可以使用 AWS Management Console、AWS CloudFormation、AWS CLI 或 CloudFront API。

Console

创建 OAC

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择 Origin access（源访问）。
3. 选择 Create control setting（创建控制设置）。
4. 在创建新 OAC 页面上，执行以下操作：
   a. 输入 OAC 的名称和（可选）描述。
   b. 在签名行为中，建议保留默认设置（签署请求（推荐））。有关更多信息，请参阅the section called “源访问控制的高级设置”。
5. 对于源类型，请选择 MediaPackage V2。
6. 选择创建。

> **Tip**
>
> 创建 OAC 后，记下名称。在以下过程中，您需要此名称。

向分配中的 MediaPackage v2 源添加 OAC

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 选择具有要向其中添加 OAC 的 MediaPackage V2 源的分配，然后选择源选项卡。
3. 选择要向其中添加 OAC 的 MediaPackage v2 源，然后选择编辑。
4. 对于源的协议，请选择仅 HTTPS。
5. 从源访问控制下拉菜单中，选择要使用的 OAC。
6. 选择保存更改。

分配开始部署到所有 CloudFront 边缘站点。当边缘站点收到新配置时，它会签署自己发送到 MediaPackage v2 源的所有请求。

CloudFormation

要使用 AWS CloudFormation 创建 OAC，请使用 `AWS::CloudFront::OriginAccessControl` 资源类型。以下示例演示了 YAML 格式的 AWS CloudFormation 模板语法，用于创建 OAC。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: mediapackagev2
      SigningBehavior: always
      SigningProtocol: sigv4
```

有关更多信息，请参阅 AWS CloudFormation 用户指南 中的 AWS::CloudFront::OriginAccessControl。

CLI

要使用 AWS Command Line Interface (AWS CLI) 创建源访问控制，请使用 aws cloudfront create-origin-access-control 命令。您可以使用输入文件来提供命令的输入参数，而不是将每个单独的参数指定为命令行输入。

创建源访问控制（带输入文件的 CLI）

1. 使用以下命令创建名为 `origin-access-control.yaml` 的文件。此文件包含 create-origin-access-control 命令的所有输入参数。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. 打开刚创建的 `origin-access-control.yaml` 文件。编辑文件以添加 OAC 的名称、描述（可选），然后将 `SigningBehavior` 更改为 `always`。然后保存文件。

   有关其他 OAC 设置的信息，请参阅the section called “源访问控制的高级设置”。

3. 使用以下命令通过 `origin-access-control.yaml` 文件中的输入参数创建源访问控制。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

   记下命令输出中的 `Id` 值。您需要使用它将 OAC 添加到 CloudFront 分配中的 MediaPackage v2 源。

将 OAC 附加到现有分配中的 MediaPackage v2 源（带输入文件的 CLI）

1. 使用以下命令保存要向其添加 OAC 的 CloudFront 分配的分配配置。该分配必须具有 MediaPackage v2 源。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. 打开刚创建的名为 `dist-config.yaml` 的文件。编辑文件，进行以下更改：

   - 在 `Origins` 对象中，将 OAC 的 ID 添加到名为 `OriginAccessControlId` 的字段中。

- 从名为 `OriginAccessIdentity` 的字段中移除值（如果存在）。
- 将 `ETag` 字段重命名为 `IfMatch`，但不更改字段的值。

完成后保存该文件。

3. 使用以下命令更新分配以使用源访问控制。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

分配开始部署到所有 CloudFront 边缘站点。当边缘站点收到新配置时，它会签署自己发送到 MediaPackage v2 源的所有请求。

API

要使用 CloudFront API 创建 OAC，请使用 CreateOriginAccessControl。有关您在此 API 调用中指定的字段的更多信息，请参阅有关 AWS SDK 或其他 API 客户端的 API 参考文档。

创建 OAC 后，可以使用下面的任何一个 API 调用将其附加到分配中的 MediaPackage v2 源：

- 要将它附加到现有分配，请使用 UpdateDistribution。
- 要将它附加到新分配，请使用 CreateDistribution。

对于这两个 API 调用，请在源内的 `OriginAccessControlId` 字段中提供 OAC ID。有关您在这些 API 调用中指定的其他字段的更多信息，请参阅 分配设置参考 以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

源访问控制的高级设置

CloudFront OAC 功能包括仅适用于特定使用案例的高级设置。除非您特别需要高级设置，否则请使用推荐的设置。

OAC 包含一个名为签名行为（在控制台中）或 `SigningBehavior`（在 API、CLI 和 AWS CloudFormation 中）的设置。此设置提供以下选项：

始终签署源请求（推荐设置）

我们建议使用此设置，此设置在控制台中名为签署请求（推荐），在 API、CLI 和 AWS CloudFormation 中名为 `always`。使用此设置，CloudFront 将始终签署自己发送到 MediaPackage v2 源的所有请求。

切勿签署源请求

此设置在控制台中名为 Do not sign requests（请勿签署请求），在 API、CLI 和 AWS CloudFormation 中名为 `never`。使用此设置关闭所有使用此 OAC 的分配中所有源的 OAC。与从所有使用 OAC 的源和分配中逐一删除 OAC 相比，这可以节省时间和精力。使用此设置，CloudFront 不会签署自己发送到 MediaPackage v2 源的任何请求。

> **Warning**
>
> 要使用此设置，MediaPackage v2 源必须可供公开访问。如果您将此设置用于不可公开访问的 MediaPackage v2 源，则 CloudFront 无法访问该源。MediaPackage v2 源向 CloudFront 返回错误，而 CloudFront 将这些错误传递给查看器。有关更多信息，请参阅《AWS Elemental MediaPackage 用户指南》中针对 MediaPackage 中的策略和权限的 MediaPackage v2 策略示例。

不要改写查看器（客户端）**`Authorization`** 标头

此设置在控制台中名为 Do not override authorization header（请勿改写授权标头），在 API、CLI 和 AWS CloudFormation 中名为 `no-override`。如果您希望 CloudFront 仅在相应的查看器请求不包含 `Authorization` 标头时签署源请求，请使用此设置。使用此设置，CloudFront 会在查看器请求中存在 `Authorization` 标头时传递该标头，但在查看器请求不包含 `Authorization` 标头时对源请求进行签名（添加自己的 `Authorization` 标头）。

> **Warning**
>
> 要传递查看器请求的 `Authorization` 标头，您必须 针对所有使用与此源访问控制关联的 MediaPackage v2 源的缓存行为将 `Authorization` 标头添加到缓存策略中。

限制对AWS Elemental MediaStore源的访问

CloudFront 提供源访问控制（OAC），用于限制对 AWS Elemental MediaStore 源的访问。

主题

创建新的源访问控制

完成以下主题中描述的步骤，在 CloudFront 中设置新的源访问控制。

主题

先决条件

在创建和设置源访问控制之前，您必须拥有带有 MediaStore 源的 CloudFront 分配。

向源访问控制授予访问 MediaStore 源的权限

在创建源访问控制或在 CloudFront 分配中设置它之前，请确保 OAC 有权访问 MediaStore 源。请在创建 CloudFront 分配后，但在分配配置中将 OAC 添加到 MediaStore 源之前，执行此操作。

要授予 OAC 访问 MediaStore 源的权限，请使用 MediaStore 容器策略，以允许 CloudFront 服务主体（`cloudfront.amazonaws.com`）访问源。使用策略中的 `Condition` 元素，仅在请求代表包含 MediaStore 源的 CloudFront 分配时，才允许 CloudFront 访问 MediaStore 容器。

以下是允许 CloudFront OAC 访问 MediaStore 源的 MediaStore 容器策略示例。

Example 允许对 CloudFront OAC 进行只读访问的 MediaStore 容器策略

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AllowCloudFrontServicePrincipalReadOnly",
                "Effect": "Allow",
                "Principal": {
                  "Service": "cloudfront.amazonaws.com"
                },
```

```
                "Action": [
                  "mediastore:GetObject"
                ],
                "Resource":
 "arn:aws:mediastore:<region>:111122223333:container/<container name>/*",
                "Condition": {
                    "StringEquals": {
                      "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                    },
                    "Bool": {
                      "aws:SecureTransport": "true"
                    }
                }
            }
        ]
}
```

Example 允许对 CloudFront OAC 进行读写访问的 MediaStore 容器策略

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "AllowCloudFrontServicePrincipalReadWrite",
                "Effect": "Allow",
                "Principal": {
                  "Service": "cloudfront.amazonaws.com"
                },
                "Action": [
                  "mediastore:GetObject",
                  "mediastore:PutObject"
                ],
                "Resource":
 "arn:aws:mediastore:<region>:111122223333:container/<container name>/*",
                "Condition": {
                    "StringEquals": {
                      "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                    },
                    "Bool": {
                      "aws:SecureTransport": "true"
                    }
```

```
                }
            }
        ]
}
```

> **Note**
>
> 要允许写访问，您必须在 CloudFront 分配的行为设置中将 Allowed HTTP methods（允许的 HTTP 方法）配置为包含 PUT。

创建源访问控制

要创建 OAC，可以使用 AWS Management Console、AWS CloudFormation、AWS CLI 或 CloudFront API。

Console

创建源访问控制

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择 Origin access（源访问）。
3. 选择 Create control setting（创建控制设置）。
4. 在 Create control setting（创建控制设置）表单上，执行以下操作：
   a. 在 Details（详细信息）窗格中，输入源访问控制的 Name（名称）和（可选）Description（描述）。
   b. 在 Settings（设置）中，建议保留默认设置 [Sign requests (recommended)（签署请求（推荐））]。有关更多信息，请参阅 the section called “源访问控制的高级设置”。
5. 从 Origin type（源类型）下拉列表中选择 MediaStore。
6. 选择创建。

   创建 OAC 后，记下 Name（名称）。在以下过程中，您需要此名称。

向分配中的 MediaStore 源添加源访问控制

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台

2. 选择具有要向其添加 OAC 的 MediaStore 源的分配，然后选择 Origins（源）选项卡。
3. 选择要向其添加 OAC 的 MediaStore 源，然后选择 Edit（编辑）。
4. 对于源的 Protocol（协议），选择 HTTPS only（仅 HTTPS）。
5. 从 Origin access control（源访问控制）下拉菜单中，选择要使用的 OAC。
6. 选择保存更改。

分配开始部署到所有 CloudFront 边缘站点。当边缘站点收到新配置时，它会签署其发送到 MediaStore 桶源的所有请求。

CloudFormation

要使用 AWS CloudFormation 创建源访问控制（OAC），请使用 `AWS::CloudFront::OriginAccessControl` 资源类型。以下示例显示了 YAML 格式的 AWS CloudFormation 模板语法，用于创建源访问控制。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: mediastore
      SigningBehavior: always
      SigningProtocol: sigv4
```

有关更多信息，请参阅 AWS CloudFormation 用户指南 中的 AWS::CloudFront::OriginAccessControl。

CLI

要使用 AWS Command Line Interface (AWS CLI) 创建源访问控制，请使用 aws cloudfront create-origin-access-control 命令。您可以使用输入文件来提供命令的输入参数，而不是将每个单独的参数指定为命令行输入。

创建源访问控制（带输入文件的 CLI）

1. 使用以下命令创建名为 `origin-access-control.yaml` 的文件。此文件包含 create-origin-access-control 命令的所有输入参数。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. 打开刚创建的 origin-access-control.yaml 文件。编辑文件以添加 OAC 的名称、描述（可选），然后将 SigningBehavior 更改为 always。然后保存文件。

   有关其他 OAC 设置的信息，请参阅the section called “源访问控制的高级设置”。

3. 使用以下命令通过 origin-access-control.yaml 文件中的输入参数创建源访问控制。

   ```
   aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
   access-control.yaml
   ```

   记下命令输出中的 Id 值。您需要使用它将 OAC 添加到 CloudFront 分配中的 MediaStore 源。

将 OAC 附加到现有分配中的 MediaStore 源（带输入文件的 CLI）

1. 使用以下命令保存要向其添加 OAC 的 CloudFront 分配的分配配置。该分配必须有 MediaStore 源。

   ```
   aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
   output yaml > dist-config.yaml
   ```

2. 打开刚创建的名为 dist-config.yaml 的文件。编辑文件，进行以下更改：

   - 在 Origins 对象中，将 OAC 的 ID 添加到名为 OriginAccessControlId 的字段中。
   - 从名为 OriginAccessIdentity 的字段中移除值（如果存在）。
   - 将 ETag 字段重命名为 IfMatch，但不更改字段的值。

   完成后保存该文件。

3. 使用以下命令更新分配以使用源访问控制。

   ```
   aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
   input-yaml file://dist-config.yaml
   ```

分配开始部署到所有 CloudFront 边缘站点。当边缘站点收到新配置时，它会签署其发送到 MediaStore 源的所有请求。

API

要使用 CloudFront API 创建源访问控制，请使用 CreateOriginAccessControl。有关您在此 API 调用中指定的字段的更多信息，请参阅 AWS 开发工具包或其他 API 客户端的 API 参考文档。

创建源访问控制后，可以使用下面的任何一个 API 调用将其附加到分配中的 MediaStore 源：

- 要将它附加到现有分配，请使用 UpdateDistribution。
- 要将它附加到新分配，请使用 CreateDistribution。

对于这两个 API 调用，请在源内的 `OriginAccessControlId` 字段中提供源访问控制 ID。有关您在这些 API 调用中指定的其他字段的更多信息，请参阅 分配设置参考 以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

源访问控制的高级设置

CloudFront 源访问控制特征包括仅适用于特定使用案例的高级设置。除非您特别需要高级设置，否则请使用推荐的设置。

源访问控制包含一个名为 Signing behavior（签名行为）（在控制台中）或 `SigningBehavior`（在 API、CLI 和 AWS CloudFormation）的设置。此设置提供以下选项：

始终签署源请求（推荐设置）

我们建议使用此设置，此设置在控制台中名为签署请求（推荐），在 API、CLI 和 AWS CloudFormation 中名为 `always`。使用此设置，CloudFront 将始终签署其发送到 MediaStore 源的所有请求。

切勿签署源请求

此设置在控制台中名为 Do not sign requests（请勿签署请求），在 API、CLI 和 AWS CloudFormation 中名为 `never`。使用此设置关闭所有使用此源访问控制的分配中所有源的源访问控制。与从所有使用源访问控制的源和分配中逐一删除源访问控制相比，这可以节省时间和精力。使用此设置，CloudFront 不会签署其发送到 MediaStore 源的任何请求。

> **Warning**
>
> 要使用此设置，MediaStore 源必须可供公开访问。如果您将此设置用于不可公开访问的 MediaStore 源，则 CloudFront 无法访问该源。MediaStore 源向 CloudFront 返回错误，而 CloudFront 将这些错误传递给查看器。有关更多信息，请参阅通过 HTTPS 进行公有读取访问的 MediaStore 容器策略示例。

不要改写查看器（客户端）**Authorization** 标头

此设置在控制台中名为 Do not override authorization header（请勿改写授权标头），在 API、CLI 和 AWS CloudFormation 中名为 `no-override`。如果您希望 CloudFront 仅在相应的查看器请求不包含 `Authorization` 标头时签署源请求，请使用此设置。使用此设置，CloudFront 会在查看器请求中存在 `Authorization` 标头时传递该标头，但在查看器请求不包含 `Authorization` 标头时对源请求进行签名（添加自己的 `Authorization` 标头）。

> **Warning**
>
> 要传递查看器请求的 `Authorization` 标头，您必须 针对所有使用与此源访问控制关联的 MediaStore 源的缓存行为将 `Authorization` 标头添加到缓存策略中。

限制对 AWS Lambda 函数 URL 源的访问

CloudFront 提供源访问控制（OAC），以限制对 Lambda 函数 URL 源的访问。

主题

- 创建新 OAC
- 源访问控制的高级设置

创建新 OAC

完成以下主题中描述的步骤，在 CloudFront 中设置新的 OAC。

> **Note**
>
> 如果您在 Lambda 函数 URL 中使用 PUT 或 POST 方法，则您的用户在向 CloudFront 发送请求时，必须在 `x-amz-content-sha256` 标头中包含有效负载哈希值。Lambda 不支持未签名的有效负载。

主题

先决条件

在创建和设置 OAC 之前，您必须拥有将 Lambda 函数 URL 作为源的 CloudFront 分配。有关更多信息，请参阅 使用 Lambda 函数 URL。

向 OAC 授予访问 Lambda 函数 URL 的权限

在创建 OAC 或在 CloudFront 分配中设置它之前，请确保 OAC 有权访问 Lambda 函数 URL。请在创建 CloudFront 分配后，但在分配配置中将 OAC 添加到 Lambda 函数 URL 之前，执行此操作。

> **Note**
>
> 要更新 Lambda 函数 URL 的 IAM 策略，您必须使用 AWS Command Line Interface（AWS CLI）。目前不支持在 Lambda 控制台中编辑 IAM 策略。

以下 AWS CLI 命令授予 CloudFront 服务主体（`cloudfront.amazonaws.com`）访问您的 Lambda 函数 URL 的权限。使用策略中的 `Condition` 元素，仅 在该请求代表包含 Lambda 函数 URL 的 CloudFront 分配时，才允许 CloudFront 访问 Lambda。

Example 示例：AWS CLI 命令更新策略以允许对 CloudFront OAC 进行只读访问

以下 AWS CLI 命令允许 CloudFront 分配（*`E1PDK09ESKHJWT`*）访问您的 Lambda *`FUNCTION_URL_NAME`*。

```
aws lambda add-permission \
```

```
--statement-id "AllowCloudFrontServicePrincipal" \
--action "lambda:InvokeFunctionUrl" \
--principal "cloudfront.amazonaws.com" \
--source-arn "arn:aws:cloudfront::123456789012:distribution/E1PDK09ESKHJWT" \
--function-name FUNCTION_URL_NAME
```

> **Note**
>
> 如果您创建了一个分配，但它没有访问您的 Lambda 函数 URL 的权限，则可以从 CloudFront 控制台中选择复制 CLI 命令，然后从命令行终端输入此命令。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的向 AWS 服务 授予函数访问权。

创建 OAC

要创建 OAC，可以使用 AWS Management Console、AWS CloudFormation、AWS CLI 或 CloudFront API。

Console

创建 OAC

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择 Origin access（源访问）。
3. 选择 Create control setting（创建控制设置）。
4. 在创建新 OAC 页面上，执行以下操作：
   a. 输入 OAC 的名称和（可选）描述。
   b. 在签名行为中，建议保留默认设置（签署请求（推荐））。有关更多信息，请参阅 the section called “源访问控制的高级设置”。
5. 对于源类型，请选择 Lambda。
6. 选择创建。

> **Tip**
>
> 创建 OAC 后，记下名称。在以下过程中，您需要此名称。

向分配中的 Lambda 函数 URL 添加源访问控制

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 选择具有要向其中添加 OAC 的 Lambda 函数 URL 的分配，然后选择源选项卡。
3. 选择要向其中添加 OAC 的 Lambda 函数 URL，然后选择编辑。
4. 对于源的协议，请选择仅 HTTPS。
5. 从源访问控制下拉菜单中，选择要使用的 OAC。
6. 选择保存更改。

分配开始部署到所有 CloudFront 边缘站点。当边缘站点收到新配置时，它会签署自己发送到 Lambda 函数 URL 的所有请求。

CloudFormation

要使用 AWS CloudFormation 创建 OAC，请使用 `AWS::CloudFront::OriginAccessControl` 资源类型。以下示例演示了 YAML 格式的 AWS CloudFormation 模板语法，用于创建 OAC。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: lambda
      SigningBehavior: always
      SigningProtocol: sigv4
```

有关更多信息，请参阅 AWS CloudFormation 用户指南 中的 AWS::CloudFront::OriginAccessControl。

CLI

要使用 AWS Command Line Interface (AWS CLI) 创建源访问控制，请使用 aws cloudfront create-origin-access-control 命令。您可以使用输入文件来提供命令的输入参数，而不是将每个单独的参数指定为命令行输入。

创建源访问控制（带输入文件的 CLI）

1. 使用以下命令创建名为 `origin-access-control.yaml` 的文件。此文件包含 create-origin-access-control 命令的所有输入参数。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. 打开刚创建的 origin-access-control.yaml 文件。编辑文件以添加 OAC 的名称、描述（可选），然后将 SigningBehavior 更改为 always。然后保存文件。

   有关其他 OAC 设置的信息，请参阅the section called “源访问控制的高级设置”。

3. 使用以下命令通过 origin-access-control.yaml 文件中的输入参数创建源访问控制。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

   记下命令输出中的 Id 值。您需要使用它将 OAC 添加到 CloudFront 分配中的 Lambda 函数 URL。

将 OAC 附加到现有分配中的 Lambda 函数 URL（带输入文件的 CLI）

1. 使用以下命令保存要向其添加 OAC 的 CloudFront 分配的分配配置。该分配必须将 Lambda 函数 URL 作为源。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. 打开刚创建的名为 dist-config.yaml 的文件。编辑文件，进行以下更改：

   - 在 Origins 对象中，将 OAC 的 ID 添加到名为 OriginAccessControlId 的字段中。
   - 从名为 OriginAccessIdentity 的字段中移除值（如果存在）。
   - 将 ETag 字段重命名为 IfMatch，但不更改字段的值。

   完成后保存该文件。

3. 使用以下命令更新分配以使用源访问控制。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

分配开始部署到所有 CloudFront 边缘站点。当边缘站点收到新配置时，它会签署自己发送到 Lambda 函数 URL 的所有请求。

API

要使用 CloudFront API 创建 OAC，请使用 CreateOriginAccessControl。有关您在此 API 调用中指定的字段的更多信息，请参阅有关 AWS SDK 或其他 API 客户端的 API 参考文档。

创建 OAC 后，可以使用下面的任何一个 API 调用将其附加到分配中的 Lambda 函数 URL：

- 要将它附加到现有分配，请使用 UpdateDistribution。
- 要将它附加到新分配，请使用 CreateDistribution。

对于这两个 API 调用，请在源内的 `OriginAccessControlId` 字段中提供 OAC ID。有关您在这些 API 调用中指定的其他字段的更多信息，请参阅有关 AWS SDK 或其他 API 客户端的 API 参考文档。

源访问控制的高级设置

CloudFront OAC 功能包括仅适用于特定使用案例的高级设置。除非您特别需要高级设置，否则请使用推荐的设置。

OAC 包含一个名为签名行为（在控制台中）或 `SigningBehavior`（在 API、CLI 和 AWS CloudFormation 中）的设置。此设置提供以下选项：

始终签署源请求（推荐设置）

我们建议使用此设置，此设置在控制台中名为签署请求（推荐），在 API、CLI 和 AWS CloudFormation 中名为 `always`。使用此设置，CloudFront 将始终签署自己发送到 Lambda 函数 URL 的所有请求。

切勿签署源请求

此设置在控制台中名为 Do not sign requests（请勿签署请求），在 API、CLI 和 AWS CloudFormation 中名为 `never`。使用此设置关闭所有使用此 OAC 的分配中所有源的 OAC。与从所有使用 OAC 的源和分配中逐一删除 OAC 相比，这可以节省时间和精力。使用此设置，CloudFront 不会签署自己发送到 Lambda 函数 URL 的任何请求。

> **Warning**
>
> 要使用此设置，Lambda 函数 URL 必须可供公开访问。如果您将此设置用于不可公开访问的 Lambda 函数 URL，则 CloudFront 无法访问该源。Lambda 函数 URL 源向 CloudFront 返回错误，而 CloudFront 会将这些错误传递给查看器。有关更多信息，请参阅《AWS Lambda 用户指南》中的 Lambda 函数 URL 的安全性和身份验证模型。

不要改写查看器（客户端）**Authorization** 标头

此设置在控制台中名为 Do not override authorization header（请勿改写授权标头），在 API、CLI 和 AWS CloudFormation 中名为 `no-override`。如果您希望 CloudFront 仅在相应的查看器请求不包含 `Authorization` 标头时签署源请求，请使用此设置。使用此设置，CloudFront 会在查看器请求中存在 `Authorization` 标头时传递该标头，但在查看器请求不包含 `Authorization` 标头时对源请求进行签名（添加自己的 `Authorization` 标头）。

> **Warning**
>
> 要传递查看器请求的 `Authorization` 标头，您必须 针对所有使用与此源访问控制关联的 Lambda 函数 URL 的缓存行为，将 `Authorization` 标头添加到缓存策略中。

限制对 Amazon Simple Storage Service 源的访问

CloudFront 提供两种向 Amazon S3 源发送经身份验证的请求的方式：源访问控制（OAC）和源访问身份 (OAI)。OAC 可帮助您保护源（如 Amazon S3 源）。建议使用 OAC，因为它支持：

- AWS 区域 中的所有 Amazon S3 存储桶，包括 2022 年 12 月之后推出的选择加入区域
- Amazon S3 使用 AWS KMS 的服务器端加密 (SSE-KMS)
- 对 Amazon S3 的动态请求（`PUT` 和 `DELETE`）

来源访问身份 (OAI) 不适用于前面列表中的场景，或者在这些场景中需要额外的解决方法。以下主题介绍了如何将源访问控制（OAC）与 Amazon S3 来源配合使用。有关如何从来源访问身份 (OAI) 迁移到源访问控制（OAC）的信息，请参阅the section called “从源访问身份 (OAI) 迁移到源访问控制（OAC）”。

> **注意**
>
> - 当您将 CloudFront OAC 与 Amazon S3 存储桶来源一起使用时，必须将 Amazon S3 对象所有权设置为强制存储桶拥有者，这是新 Amazon S3 存储桶的默认值。如果您需要 ACL，请使用存储桶拥有者优先设置来控制通过 CloudFront 上传的对象。
> - 如果您的源是配置为网站端点的 Amazon S3 存储桶，则必须使用 CloudFront 将其设置为自定义源。这意味着您无法使用 OAC（或 OAI）。OAC 不支持使用 Lambda@Edge 进行源重定向。

主题

创建新的源访问控制

完成以下主题中描述的步骤，在 CloudFront 中设置新的源访问控制。

主题

先决条件

在创建和设置源访问控制（OAC）之前，您必须拥有带有 Amazon S3 存储桶源的 CloudFront 分配。此源必须是常规 S3 存储桶，而不是配置为网站端点的存储桶。有关设置 CloudFront 分配与 S3 存储桶源配合使用的更多信息，请参阅the section called “开始使用基本分配”。

> **Note**
>
> 当您使用 OAC 保护 S3 存储桶源时，无论您的具体设置如何，CloudFront 和 Amazon S3 之间的通信始终 通过 HTTPS 进行。

向源访问控制授予访问 S3 存储桶的权限

在创建源访问控制（OAC）或在 CloudFront 分配中进行设置之前，请确保 OAC 有权访问 S3 存储桶源。请在创建 CloudFront 分配后，但在分配配置中将 OAC 添加到 S3 源之前，执行此操作。

要授予 OAC 访问 S3 存储桶的权限，请使用 S3 存储桶策略，以允许 CloudFront 服务主体 (`cloudfront.amazonaws.com`) 访问存储桶。使用策略中的 `Condition` 元素，仅在请求代表包含 S3 源的 CloudFront 分配时，才允许 CloudFront 访问存储桶。

有关添加或修改存储桶策略的信息，请参阅 Amazon S3 用户指南 中的使用 Amazon S3 控制台添加存储桶策略。

以下是允许 CloudFront OAC 访问 S3 源的 S3 存储桶策略示例。

Example 允许对 CloudFront OAC 进行只读访问的 S3 存储桶策略

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowCloudFrontServicePrincipalReadOnly",
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudfront.amazonaws.com"
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::<S3 bucket name>/*",
        "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
    }
}
```

Example 允许对 CloudFront OAC 进行读写访问的 S3 存储桶策略

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowCloudFrontServicePrincipalReadWrite",
        "Effect": "Allow",
        "Principal": {
            "Service": "cloudfront.amazonaws.com"
        },
        "Action": [
            "s3:GetObject",
            "s3:PutObject"
        ],
        "Resource": "arn:aws:s3:::<S3 bucket name>/*",
        "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
    }
}
```

SSE-KMS

如果 S3 存储桶源中的对象是通过采用 AWS Key Management Service 的服务器端加密 (SSE-KMS) 进行加密的，您必须确保 OAC 有权使用 AWS KMS 密钥。要向 OAC 授予使用 KMS 密钥的权限，请向 KMS 密钥策略添加一条语句。有关如何修改密钥策略的信息，请参阅 AWS Key Management Service 开发人员指南 中的更改密钥策略。

以下示例显示了允许 OAC 使用 KMS 密钥的 KMS 密钥策略语句。

Example 允许 CloudFront OAC 访问 SSE-KMS 的 KMS 密钥的 KMS 密钥策略语句

```
{
    "Sid": "AllowCloudFrontServicePrincipalSSE-KMS",
    "Effect": "Allow",
    "Principal": {
        "Service": [
            "cloudfront.amazonaws.com"
        ]
```

```
     },
    "Action": [
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*",
    "Condition": {
            "StringEquals": {
                "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
            }
        }
}
```

创建源访问控制

要创建源访问控制（OAC），可以使用 AWS Management Console、AWS CloudFormation、AWS CLI 或 CloudFront API。

Console

创建源访问控制

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择 Origin access（源访问）。
3. 选择 Create control setting（创建控制设置）。
4. 在 Create control setting（创建控制设置）表单上，执行以下操作：
   a. 在 Details（详细信息）窗格中，输入源访问控制的 Name（名称）和（可选）Description（描述）。
   b. 在 Settings（设置）中，建议保留默认设置 [Sign requests (recommended)（签署请求（推荐））]。有关更多信息，请参阅 the section called “源访问控制的高级设置”。
5. 从 Origin type（源类型）下拉列表中选择 S3。
6. 选择创建。

   创建 OAC 后，记下 Name（名称）。在以下过程中，您需要此名称。

向分配中的 S3 源添加源访问控制

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 选择具有要向其添加 OAC 的 S3 源的分配，然后选择 Origins（源）选项卡。
3. 选择要向其添加 OAC 的 S3 源，然后选择编辑。
4. 在源访问部分，选择源访问控制设置（推荐）。
5. 从源访问控制下拉菜单中，选择要使用的 OAC。
6. 选择保存更改。

分配开始部署到所有 CloudFront 边缘站点。当边缘站点收到新配置时，它会签署其发送到 S3 存储桶源的所有请求。

CloudFormation

要使用 AWS CloudFormation 创建源访问控制（OAC），请使用 `AWS::CloudFront::OriginAccessControl` 资源类型。以下示例显示了 YAML 格式的 AWS CloudFormation 模板语法，用于创建源访问控制。

```
Type: AWS::CloudFront::OriginAccessControl
Properties:
  OriginAccessControlConfig:
      Description: An optional description for the origin access control
      Name: ExampleOAC
      OriginAccessControlOriginType: s3
      SigningBehavior: always
      SigningProtocol: sigv4
```

有关更多信息，请参阅 AWS CloudFormation 用户指南 中的 AWS::CloudFront::OriginAccessControl。

CLI

要使用 AWS Command Line Interface (AWS CLI) 创建源访问控制，请使用 aws cloudfront create-origin-access-control 命令。您可以使用输入文件来提供命令的输入参数，而不是将每个单独的参数指定为命令行输入。

创建源访问控制（带输入文件的 CLI）

1. 使用以下命令创建名为 `origin-access-control.yaml` 的文件。此文件包含 create-origin-access-control 命令的所有输入参数。

```
aws cloudfront create-origin-access-control --generate-cli-skeleton yaml-input >
 origin-access-control.yaml
```

2. 打开刚创建的 origin-access-control.yaml 文件。编辑文件以添加 OAC 的名称、描述（可选），然后将 SigningBehavior 更改为 always。然后保存文件。

   有关其他 OAC 设置的信息，请参阅the section called “源访问控制的高级设置”。

3. 使用以下命令通过 origin-access-control.yaml 文件中的输入参数创建源访问控制。

```
aws cloudfront create-origin-access-control --cli-input-yaml file://origin-
access-control.yaml
```

   记下命令输出中的 Id 值。您需要使用它将 OAC 添加到 CloudFront 分配中的 S3 存储桶源。

将 OAC 附加到现有分配中的 S3 存储桶源（带输入文件的 CLI）

1. 使用以下命令保存要向其添加 OAC 的 CloudFront 分配的分配配置。该分配必须有 S3 存储桶源。

```
aws cloudfront get-distribution-config --id <CloudFront distribution ID> --
output yaml > dist-config.yaml
```

2. 打开刚创建的名为 dist-config.yaml 的文件。编辑文件，进行以下更改：

   - 在 Origins 对象中，将 OAC 的 ID 添加到名为 OriginAccessControlId 的字段中。
   - 从名为 OriginAccessIdentity 的字段中移除值（如果存在）。
   - 将 ETag 字段重命名为 IfMatch，但不更改字段的值。

   完成后保存该文件。

3. 使用以下命令更新分配以使用源访问控制。

```
aws cloudfront update-distribution --id <CloudFront distribution ID> --cli-
input-yaml file://dist-config.yaml
```

分配开始部署到所有 CloudFront 边缘站点。当边缘站点收到新配置时，它会签署其发送到 S3 存储桶源的所有请求。

API

要使用 CloudFront API 创建源访问控制，请使用 CreateOriginAccessControl。有关您在此 API 调用中指定的字段的更多信息，请参阅 AWS 开发工具包或其他 API 客户端的 API 参考文档。

创建源访问控制后，可以使用下面的任何一个 API 调用将其附加到分配中的 S3 存储桶源：

- 要将它附加到现有分配，请使用 UpdateDistribution。
- 要将它附加到新分配，请使用 CreateDistribution。

对于这两个 API 调用，请在源内的 `OriginAccessControlId` 字段中提供源访问控制 ID。有关您在这些 API 调用中指定的其他字段的更多信息，请参阅 分配设置参考 以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

删除将其 OAC 附加到 S3 存储桶的分配

如果您需要删除将其 OAC 附加到 S3 存储桶的分配，则应首先删除该分配，然后再删除 S3 存储桶源。或者，在源域名中包含该区域。如果无法做到这一点，则可以在删除之前切换到公共模式以便从分配中删除 OAC。有关更多信息，请参阅 删除分配。

从源访问身份 (OAI) 迁移到源访问控制（OAC）

要从旧式源访问身份 (OAI) 迁移到源访问控制（OAC），请先更新 S3 存储桶源以允许 OAI 和 OAC 访问存储桶的内容。这可确保 CloudFront 在过渡期间不会丢失对存储桶的访问权限。要允许 OAI 和 OAC 访问 S3 存储桶，请更新存储桶策略以包括两个语句，每种主体一个。

以下示例 S3 存储桶策略允许 OAI 和 OAC 访问 S3 源。

Example 允许对 OAI 和 OAC 进行只读访问的 S3 存储桶策略

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCloudFrontServicePrincipalReadOnly",
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
```

```
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/<CloudFront distribution ID>"
                }
            }
        },
        {
            "Sid": "AllowLegacyOAIReadOnly",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

在更新 S3 源的存储桶策略以允许访问 OAI 和 OAC 之后，您可以更新分配配置以使用 OAC（而不是 OAI）。有关更多信息，请参阅 the section called “创建新的源访问控制”。

完全部署分配后，您可以删除存储桶策略中允许访问 OAI 的语句。有关更多信息，请参阅 the section called “向源访问控制授予访问 S3 存储桶的权限”。

源访问控制的高级设置

CloudFront 源访问控制特征包括仅适用于特定使用案例的高级设置。除非您特别需要高级设置，否则请使用推荐的设置。

源访问控制包含一个名为 Signing behavior（签名行为）（在控制台中）或 `SigningBehavior`（在 API、CLI 和 AWS CloudFormation）的设置。此设置提供以下选项：

始终签署源请求（推荐设置）

> 我们建议使用此设置，此设置在控制台中名为 Sign requests (recommended) [签署请求（推荐）]，在 API、CLI 和 AWS CloudFormation 中名为 `always`。使用此设置，CloudFront 将始终签署其发送到 S3 存储桶源的所有请求。

切勿签署源请求

此设置在控制台中名为 Do not sign requests（请勿签署请求），在 API、CLI 和 AWS CloudFormation 中名为 `never`。使用此设置关闭所有使用此源访问控制的分配中所有源的源访问控制。与从所有使用源访问控制的源和分配中逐一删除源访问控制相比，这可以节省时间和精力。使用此设置，CloudFront 不会签署其发送到 S3 存储桶源的任何请求。

> **Warning**
>
> 要使用此设置，S3 存储桶源必须可公开访问。如果您将此设置用于不可公开访问的 S3 存储桶源，则 CloudFront 无法访问该源。S3 存储桶源向 CloudFront 返回错误，而 CloudFront 会将这些错误传递给查看器。

不要改写查看器（客户端）**`Authorization`** 标头

此设置在控制台中名为 Do not override authorization header（请勿改写授权标头），在 API、CLI 和 AWS CloudFormation 中名为 `no-override`。如果您希望 CloudFront 仅在相应的查看器请求不包含 `Authorization` 标头时签署源请求，请使用此设置。使用此设置，CloudFront 会在查看器请求中存在 `Authorization` 标头时传递该标头，但在查看器请求不包含 `Authorization` 标头时对源请求进行签名（添加自己的 `Authorization` 标头）。

> **Warning**
>
> 要传递查看器请求的 `Authorization` 标头，您必须 针对所有使用与此源访问控制关联的 S3 存储桶源的缓存行为将 `Authorization` 标头添加到缓存策略中。

使用源访问身份（旧版，不推荐）

源访问身份概述

CloudFront 源访问身份 (OAI) 提供与源访问控制（OAC）类似的功能，但它并不适用于所有场景。这就是为什么我们建议改用 OAC 的原因。具体而言，OAI 不支持：

- 所有 AWS 区域中的 Amazon S3 存储桶，包括可选择加入的区域
- Amazon S3 使用 AWS KMS 的服务器端加密 (SSE-KMS)
- 对 Amazon S3 的动态请求（`PUT`、`POST` 或 `DELETE`）

- 2022 年 12 月之后推出的新 AWS 区域

有关如何从 OAI 迁移到 OAC 的信息，请参阅the section called “从源访问身份 (OAI) 迁移到源访问控制（OAC）”。

授予源访问身份读取 Amazon S3 存储桶中文件的权限

当您使用 CloudFront 控制台创建 OAI 或将 OAI 添加到分配时，可以自动更新 Amazon S3 存储桶策略以向 OAI 提供访问存储桶的权限。您也可以选择手动创建或更新存储桶策略。无论使用哪种方法，您仍应查看权限，从而确保：

- 您的 CloudFront OAI 可以代表通过 CloudFront 发出请求的查看器来访问存储桶中的文件。
- 查看器不能使用 Amazon S3 URL 访问位于 CloudFront 外部的文件。

> **Important**
>
> 如果您将 CloudFront 配置为接受并转发 CloudFront 支持的所有 HTTP 方法，请确保为您的 CloudFront OAI 授予所需的权限。例如，假设您将 CloudFront 配置为接受和转发使用 `DELETE` 方法的请求，则配置您的存储桶策略以适当地处理 `DELETE` 请求，从而确保查看器只能删除您希望它们删除的文件。

使用 Amazon S3 存储桶策略

您可以通过以下方式创建或更新 Amazon S3 存储桶策略，向 CloudFront OAI 授予对该存储桶中文件的访问权限：

- 使用 Amazon S3 控制台中 Amazon S3 存储桶的 Permissions（权限）选项卡。
- 使用 Amazon S3 API 中的 PutBucketPolicy。
- 使用 CloudFront 控制台。在 CloudFront 控制台中向源设置添加 OAI 时，您可以选择 Yes, update the bucket policy（是，更新存储桶策略），从而让 CloudFront 代表您更新存储桶策略。

如果您手动更新存储桶策略，请务必：

- 在策略中指定正确的 OAI 为 `Principal`。
- 授予 OAI 所需的权限以便代表查看器访问对象。

有关更多信息，请参阅以下部分。

在存储桶策略中将 OAI 指定为 **`Principal`**

要在 Amazon S3 存储桶策略中将 OAI 指定为 `Principal`，请使用 OAI 的 Amazon 资源名称 (ARN)，其中包括 OAI 的 ID。例如：

```
"Principal": {
    "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access Identity <origin
 access identity ID>"
}
```

在 CloudFront 控制台的安全、源访问、身份（遗留）下找到 OAI ID。或者，在 CloudFront API 中使用 ListCloudFrontOriginAccessIdentities。

向 OAI 授予权限

要向 OAI 授予对 Amazon S3 存储桶中对象的访问权限，请使用策略中与特定 Amazon S3 API 操作相关的操作。例如，`s3:GetObject` 操作允许 OAI 读取存储桶中的对象。有关更多信息，请参阅下一部分的示例，或者参阅《Amazon Simple Storage Service 用户指南》中的 Amazon S3 操作。

Amazon S3 存储桶策略示例

以下示例显示了允许 CloudFront OAI 访问 S3 存储桶的 Amazon S3 存储桶策略。

在 CloudFront 控制台的安全、源访问、身份（遗留）下找到 OAI ID。或者，在 CloudFront API 中使用 ListCloudFrontOriginAccessIdentities。

Example 向 OAI 授予读取访问权限的 Amazon S3 存储桶策略

下面的示例允许 OAI 读取指定存储桶 (`s3:GetObject`) 中的对象。

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
```

```
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

Example 向 OAI 授予读写访问权限的 Amazon S3 存储桶策略

下面的示例允许 OAI 读取和写入指定存储桶（`s3:GetObject` 和 `s3:PutObject`）中的对象。这允许查看器通过 CloudFront 将文件上传到您的 Amazon S3 存储桶。

```
{
    "Version": "2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity <origin access identity ID>"
            },
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::<S3 bucket name>/*"
        }
    ]
}
```

使用 Amazon S3 对象 ACL（不建议）

> **Important**
>
> 我们建议使用 Amazon S3 存储桶策略向 OAI 提供对 S3 存储桶的访问权限。您也可以按照这一部分的描述使用访问控制列表 (ACL)，但我们不建议您这样做。
>
> Amazon S3 建议将 bS3 Object Ownership（S3 对象所有权）设置为 bucket owner enforced（强制执行存储桶拥有者），这意味着存储桶以及其中的对象禁用 ACL。当您应用此对象所有权设置后，必须使用存储桶策略向 OAI 授予访问权限（请参阅上一部分）。
>
> 以下部分仅适用于需要 ACL 的传统使用案例。

您可以通过以下方式创建或更新文件的 ACL，向 CloudFront OAI 授予对 Amazon S3 存储桶中文件的访问权限：

- 使用 Amazon S3 控制台中 Amazon S3 对象的 Permissions（权限）选项卡。
- 使用 Amazon S3 API 中的 PutObjectAcl。

使用 ACL 向 OAI 授予访问权限时，您必须使用其 Amazon S3 规范用户 ID 来指定 OAI。在 CloudFront 控制台中，您可以在安全、源访问、身份（遗留）下找到此 ID。如果使用 CloudFront API，请使用您在创建 OAI 时返回的 `S3CanonicalUserId` 元素值，或者调用 CloudFront API 中的 ListCloudFrontOriginAccessIdentities。

在仅支持签名版本 4 身份验证的 Amazon S3 区域中使用源访问身份

较新的 Amazon S3 区域要求对通过身份验证的请求使用签名版本 4。（有关各个 Amazon S3 区域支持的签名版本，请参阅《AWS 一般参考》中的 Amazon Simple Storage Service 端点和限额。）如果您使用源访问身份并且存储桶位于要求使用签名版本 4 的区域之一，请注意以下几点：

- 支持 `DELETE`、`GET`、`HEAD`、`OPTIONS` 和 `PATCH` 请求，无限定条件。
- 不支持 `POST` 请求。

限制访问应用程序负载均衡器

对于 Web 应用程序或由 Elastic Load Balancing 中面向互联网的应用程序负载均衡器提供的其他内容，CloudFront 可以缓存对象并将它们直接提供给用户（查看者），从而减轻应用程序负载均衡器的负载。面向互联网的负载均衡器具有可公开解析的 DNS 名称，并通过互联网将来自客户端的请求路由到目标。

CloudFront 还可以帮助减少延迟，甚至吸收一些分布式拒绝服务 (DDoS) 攻击。

但是，如果用户可以绕过 CloudFront 并直接访问应用程序负载均衡器，则无法获得这些益处。但是，您可以配置 Amazon CloudFront 和应用程序负载均衡器，以防止用户直接访问应用程序负载均衡器。这使得用户只能通过 CloudFront 访问应用程序负载均衡器，从而确保您获得使用 CloudFront 的益处。

要防止用户直接访问应用程序负载均衡器并仅允许通过 CloudFront 进行访问，请完成以下高级步骤：

1. 将 CloudFront 配置为将自定义 HTTP 标头添加到向应用程序负载均衡器发送的请求中。
2. 将应用程序负载均衡器配置为仅转发包含自定义 HTTP 标头的请求。

3. （可选）需要 HTTPS 来提高此解决方案的安全性。

有关更多信息，请参阅以下主题。完成这些步骤后，用户只能通过 CloudFront 访问您的应用程序负载均衡器。

主题

配置 CloudFront 以便向请求添加自定义 HTTP 标头

您可以将 CloudFront 配置为将自定义 HTTP 标头添加到向您的源（在本例中为应用程序负载均衡器）发送的请求中。

> **Important**
>
> 此使用案例依赖于对自定义标头名称和值保密。如果标头名称和值没有保密，其他 HTTP 客户端可能会将它们包含在直接发送到应用程序负载均衡器的请求中。这可能会导致应用程序负载均衡器的行为看起来就好像请求来自 CloudFront，但实际上请求并非来自 CloudFront。为防止这种情况，请将自定义标头名称和值保密。

您可以将 CloudFront 配置为使用 CloudFront 控制台、AWS CloudFormation 或 CloudFront API 向源请求添加自定义 HTTP 标头。

要添加自定义 HTTP 标头（CloudFront 控制台）

在 CloudFront 控制台中，使用源设置中的源自定义标题设置。输入标头名称及其值，如以下示例所示。

> **Note**
>
> 本示例中的标头名称和值仅用于演示。在生产中，使用随机生成的值。将标题名称和值视为安全凭证，如用户名和密码。

当您为现有的 CloudFront 分配创建或编辑源时，以及当您创建新分配时，您可以编辑源自定义标头设置。有关更多信息，请参阅更新分配和创建分配。

要添加自定义 HTTP 标头 (AWS CloudFormation)

在 AWS CloudFormation 模板中，使用 `OriginCustomHeaders` 属性，如以下示例所示。

> **Note**
>
> 本示例中的标头名称和值仅用于演示。在生产中，使用随机生成的值。将标题名称和值视为安全凭证，如用户名和密码。

```
AWSTemplateFormatVersion: '2010-09-09'
Resources:
  TestDistribution:
    Type: 'AWS::CloudFront::Distribution'
    Properties:
      DistributionConfig:
        Origins:
          - DomainName: app-load-balancer.example.com
            Id: Example-ALB
            CustomOriginConfig:
              OriginProtocolPolicy: https-only
              OriginSSLProtocols:
                - TLSv1.2
            OriginCustomHeaders:
               - HeaderName: X-Custom-Header
                 HeaderValue: random-value-1234567890
        Enabled: 'true'
        DefaultCacheBehavior:
          TargetOriginId: Example-ALB
          ViewerProtocolPolicy: allow-all
          CachePolicyId: 658327ea-f89d-4fab-a63d-7e88639e58f6
        PriceClass: PriceClass_All
        ViewerCertificate:
          CloudFrontDefaultCertificate: 'true'
```

有关更多信息，请参阅《AWS CloudFormation 用户指南》中的 Origin 和 OriginCustomHeader 属性。

添加自定义 HTTP 标头（CloudFront API）

在 CloudFront API 中，使用 `Origin` 中的 `CustomHeaders` 对象。有关更多信息，请参阅《Amazon CloudFront API 参考》中的 CreateDistribution 和 UpdateDistribution，以及有关您的 SDK 或其他 API 客户端的文档。

有些标头名称不能指定为源自定义标头。有关更多信息，请参阅 CloudFront 无法添加到源请求的自定义标头。

将应用程序负载均衡器配置为仅转发包含特定标头的请求

将 CloudFront 配置为将自定义 HTTP 标头添加到向应用程序负载均衡器发送的请求中后（请参阅上一部分），您可以将负载均衡器配置为仅转发包含此自定义标头的请求。您可以通过添加新规则并在负载均衡器的侦听器中修改默认规则来完成此操作。

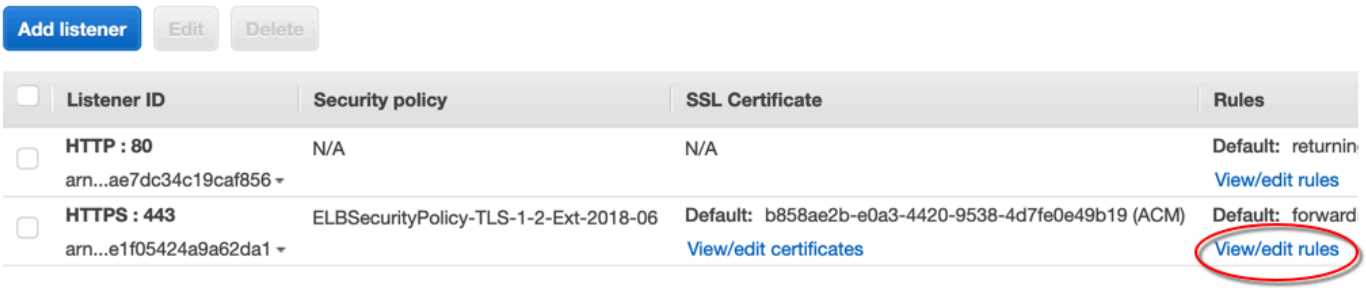
先决条件

要使用以下过程，您需要至少具有一个侦听器的应用程序负载均衡器。如果您尚未创建应用程序负载均衡器，请参阅《应用程序负载均衡器用户指南》中的创建应用程序负载均衡器。

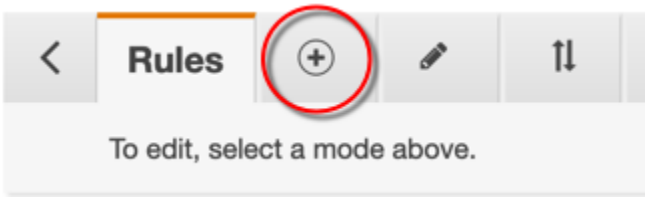
以下过程将修改 HTTPS 侦听器。您可以使用相同的过程来修改 HTTP 侦听器。

在应用程序负载均衡器侦听器中更新规则

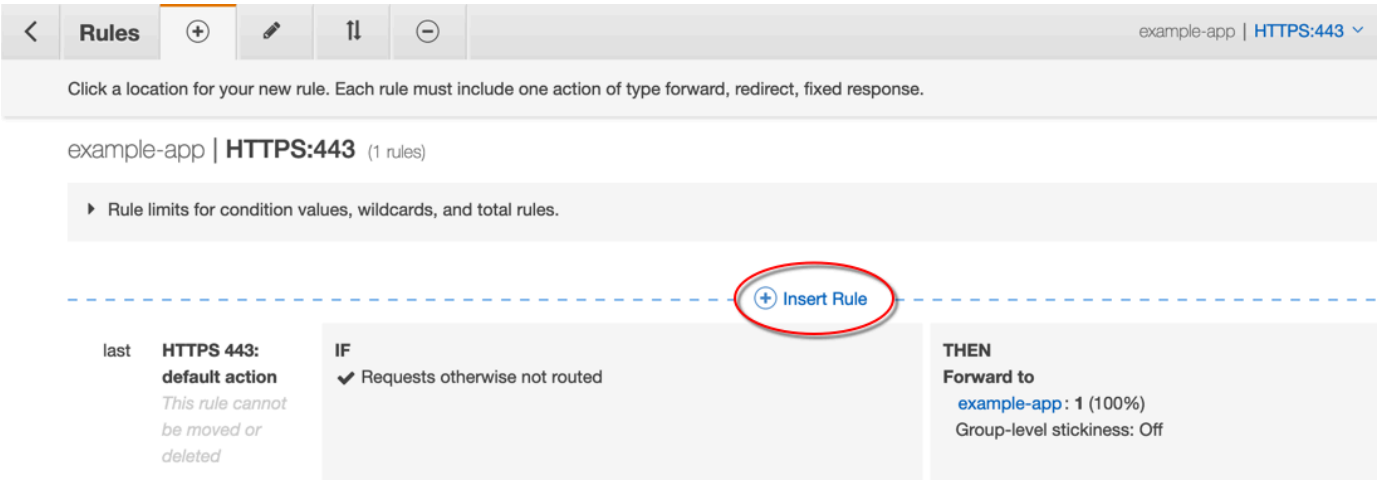
1. 在 Amazon EC2 控制台中打开负载均衡器页面。
2. 选择作为 CloudFront 分配源的负载均衡器，然后选择侦听器选项卡。
3. 对于正在修改的侦听器，请选择查看/编辑规则。



4. 选择图标以添加规则。



5. 选择插入规则。

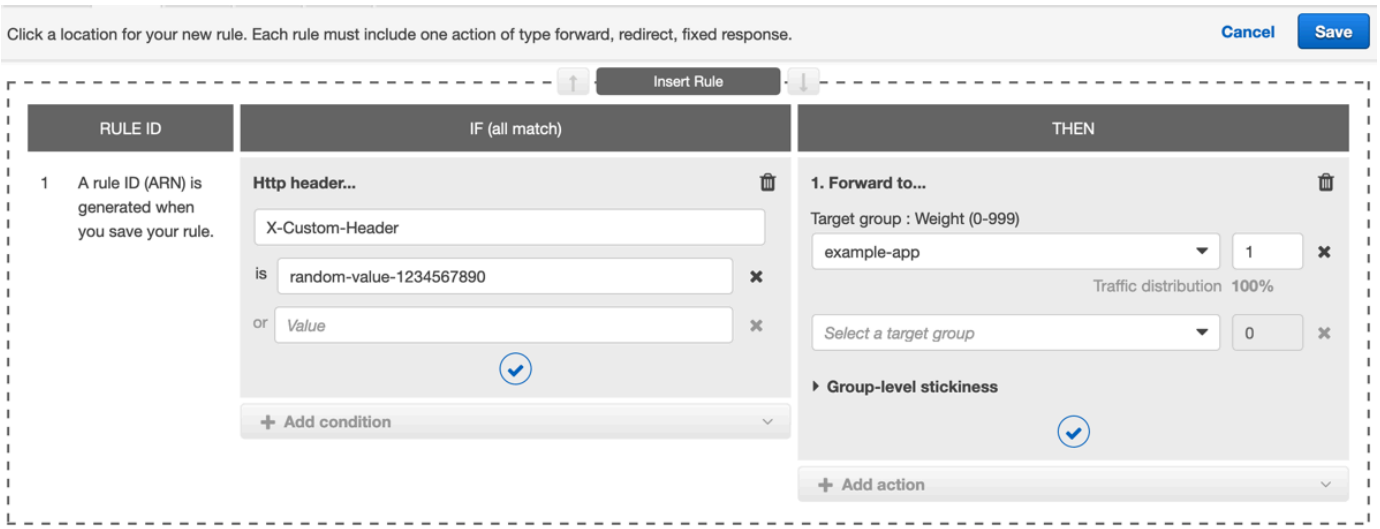


6. 对于新规则，请执行以下操作：

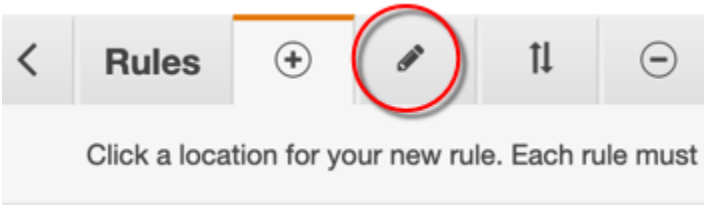
   a. 选择添加条件，然后选择 Http 标头。在 CloudFront 中指定作为源自定义标头添加的 HTTP 标头名称和值。

   b. 选择添加操作，然后选择转发到。选择要转发请求的目标组。

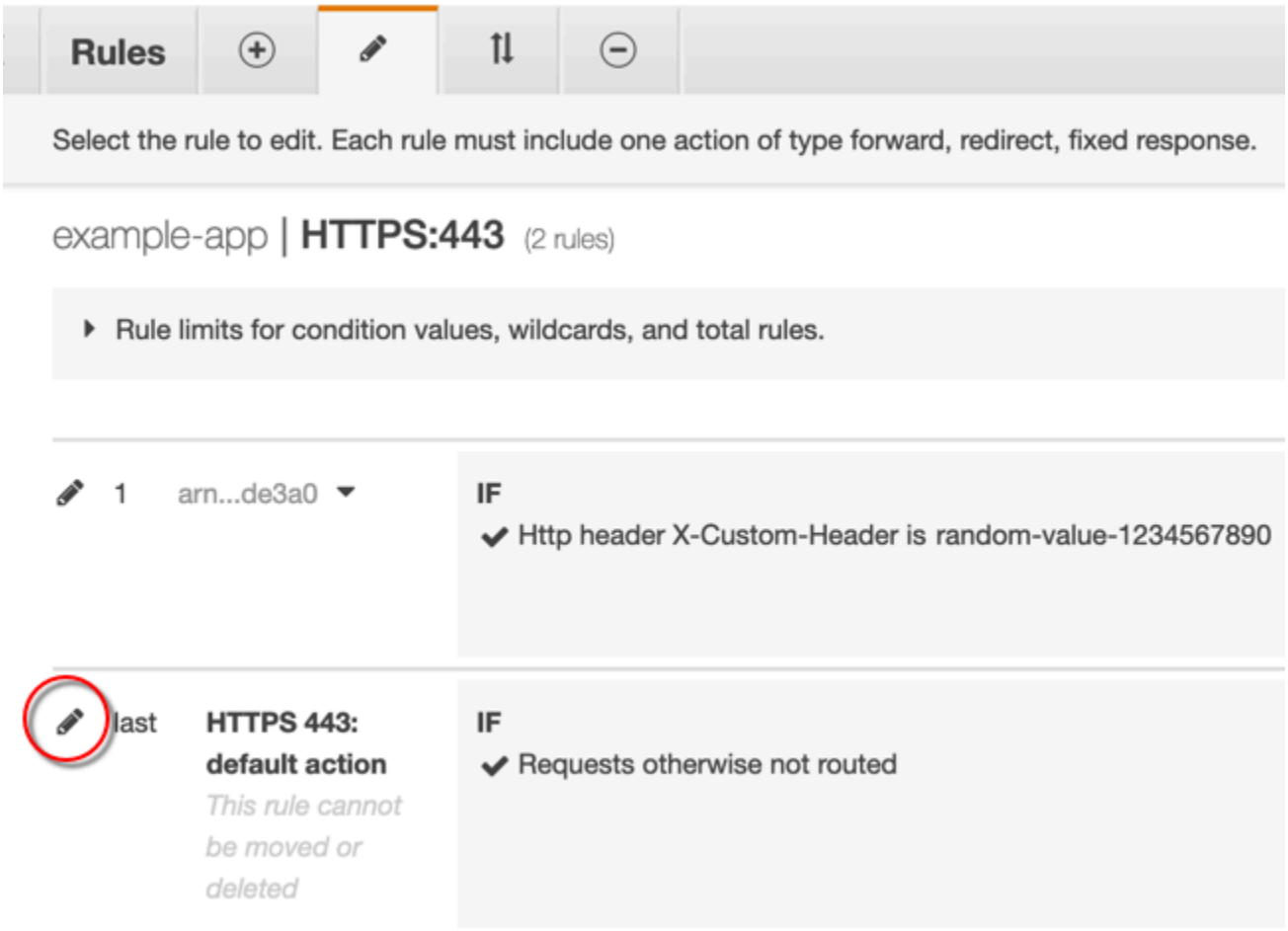
   c. 选择保存以创建新规则。



7. 选择图标以编辑规则。



8. 选择默认规则的编辑图标。



9. 对于默认规则，请执行以下操作：

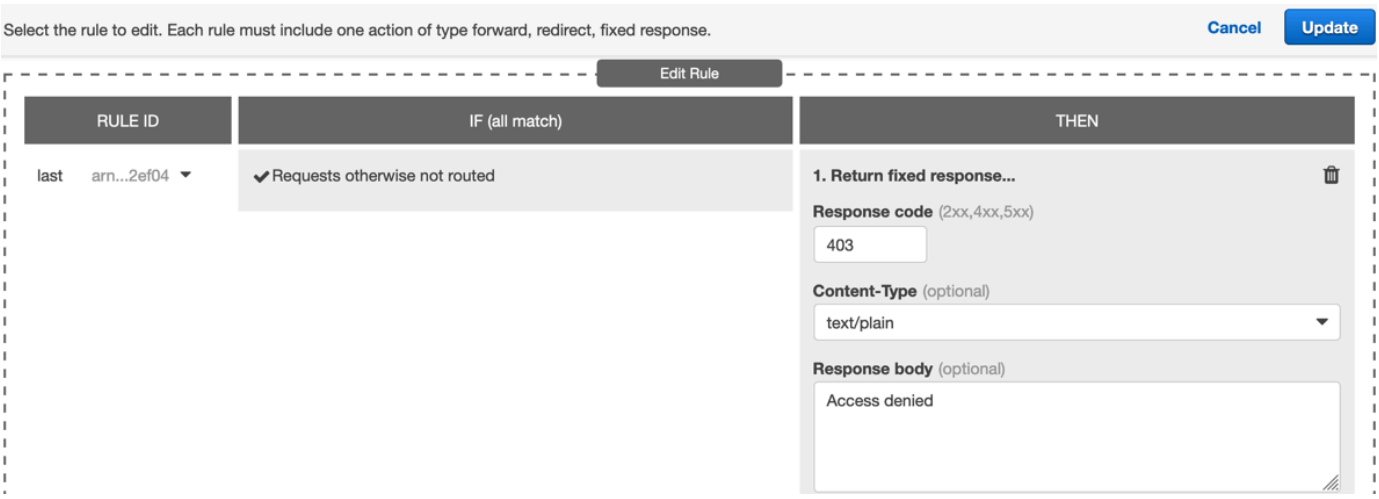
   a. 删除默认操作。

   b. 选择添加操作，然后选择返回固定响应。

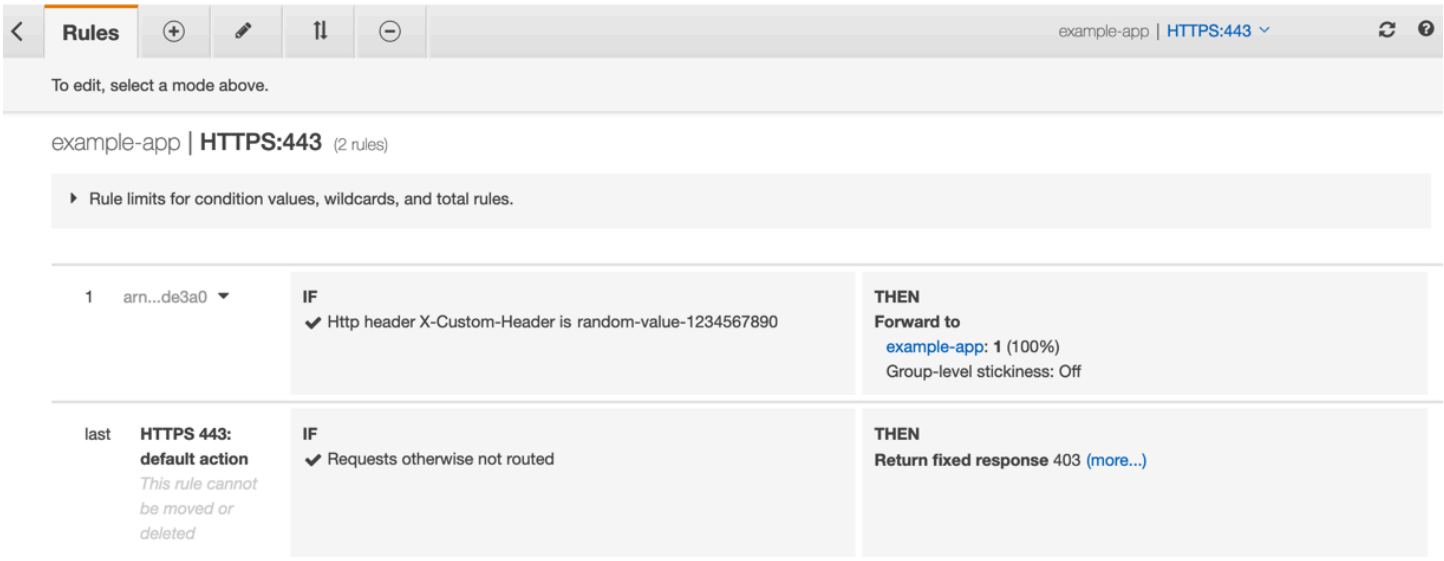
c. 对于响应代码，输入 **403**。

d. 对于响应正文，输入 **Access denied**。

e. 选择更新以更新默认规则。



完成这些步骤后，负载均衡器侦听器会有两个规则，如下图所示。第一条规则转发包含 HTTP 标头的请求（来自 CloudFront 的请求）。第二条规则向所有其他请求（并非来自 CloudFront 的请求）发送固定响应。



您可以通过向 CloudFront 分配发送请求和向应用程序负载均衡器发送一个请求来验证该解决方案是否有效。对 CloudFront 的请求将返回您的 Web 应用程序或内容，直接发送到应用程序负载均衡器的请求将返回403带有纯文本消息的响应Access denied。

（可选）提高此解决方案的安全性

为了提高此解决方案的安全性，您可以将 CloudFront 分配配置为在向应用程序负载均衡器发送请求时始终使用 HTTPS。请记住，此解决方案仅在您对自定义标头名称和值保密时才有效。使用 HTTPS 可以帮助防止窃听者发现标头名称和值。我们还建议定期轮换标头名称和值。

使用 HTTPS 进行源请求

要将 CloudFront 配置为使用 HTTPS 处理源请求，请将源协议策略设置设为仅限 HTTPS。此设置可在 CloudFront 控制台、AWS CloudFormation 和 CloudFront API 中使用。有关更多信息，请参阅 协议（仅自定义源）。

当您将 CloudFront 配置为使用 HTTPS 处理源请求时，以下规则也适用：

- 您必须将 CloudFront 配置为使用源请求策略将 `Host` 标头转发到源。您可以使用 AllViewer 托管源请求策略。
- 确保应用程序负载均衡器具有 HTTPS 侦听器（如上一节中所示）。有关更多信息，请参阅《应用程序负载均衡器用户指南》中的创建 HTTPS 侦听器。使用 HTTPS 侦听器要求您拥有与路由到应用程序负载均衡器的域名匹配的 SSL/TLS 证书。
- 只能在 AWS Certificate Manager（ACM）中的 `us-east-1` AWS 区域 中请求（或导入）CloudFront 的 SSL/TLS 证书。由于 CloudFront 是一项全球服务，因此它会自动将证书从 `us-east-1` 区域分发到与您的 CloudFront 分配关联的所有区域。
  - 例如，如果您在 `ap-southeast-2` 区域拥有应用程序负载均衡器（ALB），则必须在 `ap-southeast-2` 区域（用于在 CloudFront 和 ALB 源之间使用 HTTPS）和 `us-east-1` 区域（用于在查看器和 CloudFront 之间使用 HTTPS）中配置 SSL/TLS 证书。这两个证书都应与路由到应用程序负载均衡器的域名匹配。有关更多信息，请参阅 用于 AWS Certificate Manager 的 AWS 区域。
- 如果 Web 应用程序的最终用户（也称为查看器或客户端）可以使用 HTTPS，则还可以将 CloudFront 配置为首选（甚至需要）来自最终用户的 HTTPS 连接。为此，请使用查看器协议策略设置。您可以将其设置为将最终用户从 HTTP 重定向到 HTTPS，或拒绝使用 HTTP 的请求。此设置可在 CloudFront 控制台、AWS CloudFormation 和 CloudFront API 中使用。有关更多信息，请参阅 查看器协议策略。

轮换标头名称和值

除了使用 HTTPS 之外，还建议定期轮换标头名称和值。执行此操作的高级步骤如下：

1. 将 CloudFront 配置为将另一个自定义 HTTP 标头添加到向应用程序负载均衡器发送的请求中。

2. 更新应用程序负载均衡器侦听器规则，以转发包含另一个自定义 HTTP 标头的请求。
3. 配置 CloudFront 以停止将原始自定义 HTTP 标头添加到向应用程序负载均衡器发送的请求中。
4. 更新应用程序负载均衡器侦听器规则，以停止转发包含原始自定义 HTTP 标头的请求。

有关完成这些步骤的更多信息，请参阅前面的部分。

（可选）使用 CloudFront 的 AWS 托管前缀列表限制对源的访问

要进一步限制对应用程序负载均衡器的访问，您可以配置与应用程序负载均衡器关联的安全组，使其在服务使用 AWS 托管前缀列表时仅接受来自 CloudFront 的流量。这可以防止来自 CloudFront 以外的流量到达应用程序负载均衡器的网络层（第 3 层）或传输层（第 4 层）。

有关更多信息，请参阅 Limit access to your origins using the AWS-managed prefix list for Amazon CloudFront 博客文章。

限制您的内容的地理分配

您可以使用地理限制（有时称为地理阻止）来禁止特定地理位置的用户访问您通过 Amazon CloudFront 分配分发的内容。要使用地理限制功能，您有两个选项：

- 使用 CloudFront 地理限制功能。使用该选项可限制对与某个分配关联的所有文件的访问，并在国家/地区级别限制访问。
- 使用第三方地理定位服务。使用该选项可限制对与某个分配关联的一小部分文件的访问，或在比国家/地区级别更细化级别上限制访问。

主题

使用 CloudFront 地理限制

用户请求您的内容时，无论其处于何处，CloudFront 通常都会提供所请求的内容。如果您要阻止特定国家/地区的用户访问您的内容，可以使用 CloudFront 地理限制功能执行以下任一操作：

- 仅当用户位于允许列表中的某个已批准国家/地区时，才授予他们访问您的内容的权限。

- 当用户位于拒绝列表中被禁止的国家/地区之一时，阻止他们访问您的内容。

例如，如果一个请求来自您无权分发内容的国家/地区，您可以使用 CloudFront 地理限制功能来阻止该请求。

> **Note**
>
> CloudFront 使用第三方数据库确定用户的位置。IP 地址和国家/地区之间映射的准确性因区域而异。根据最近的测试，整体准确性为 99.8%。如果 CloudFront 无法确定用户的位置，则它会提供用户已请求的内容。

以下是地理限制的工作机制：

1. 假设您仅有权在列支敦士登分发内容。您可以更新您的 CloudFront 分配以添加仅包含列支敦士登的允许列表。（您也可以添加包含除列支敦士登以外所有国家/地区的拒绝列表。）
2. 摩纳哥的一名用户请求您的内容，DNS 将该请求路由至意大利米兰的某个 CloudFront 边缘站点。
3. 米兰的边缘站点将检查您的分配，并确定摩纳哥的这名用户没有权限，无法下载您的内容。
4. CloudFront 向该用户返回 HTTP 状态代码 `403 (Forbidden)`。

您可以选择性地将 CloudFront 配置为向该用户返回自定义错误消息，并且可以指定您希望 CloudFront 缓存请求的文件的错误响应的时间。默认值为 10 秒。有关更多信息，请参阅 为特定 HTTP 状态代码创建自定义错误页面。

地理限制适用于整个分配。如果您需要对部分内容应用一个限制，而对另一部分内容应用不同的限制（或无限制），则必须创建单独的 CloudFront 分配或使用第三方地理定位服务。

如果您启用 CloudFront 标准日志（访问日志），则可通过在日志条目中搜索其 `sc-status`（HTTP 状态代码）值为 `403` 的条目来确定 CloudFront 已拒绝的请求。不过，如果仅使用标准日志，则无法区分 CloudFront 根据用户位置拒绝的请求与 CloudFront 由于其他原因而使用户无权访问文件因而拒绝的请求。如果您有第三方地理定位服务 (例如 Digital Element 或 MaxMind)，则可基于访问日志中的 `c-ip` (客户端 IP) 列的 IP 地址来识别请求的位置。有关 CloudFront 标准日志的更多信息，请参阅配置和使用标准日志（访问日志）。

以下过程介绍如何使用 CloudFront 控制台将地理限制添加到现有的分配。有关如何使用控制台创建分配的信息，请参阅创建分配。

将地理限制添加到您的 CloudFront Web 分配（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择分配，然后选择要更新的分配。
3. 选择安全选项卡，然后选择地理限制。
4. 选择编辑。
5. 选择允许列表以创建允许的国家/地区的列表，或选择阻止列表以创建被阻止的国家/地区的列表。
6. 将所需的国家/地区添加到列表中，然后选择保存更改。

使用第三方地理定位服务

利用 CloudFront 地理限制功能，您可以在国家/地区级别控制您通过给定 Web 分配来分配的所有文件的内容分配。如果您有一个地理限制使用案例，而这些限制不遵循国家/地区界限，或者您希望限制仅可访问由给定分配提供的某些文件，则可以将 CloudFront 与第三方地理定位服务结合使用。这使您不仅可以根据国家/地区，还可以根据城市、邮政编码甚至纬度和经度来控制您的内容。

如果您使用的是第三方地理定位服务，建议您使用 CloudFront 签名的 URL，通过它可以指定过期日期和时间，在该日期和时间后，URL 不再有效。此外，建议您使用 Amazon S3 存储桶作为您的源，因为您随后可以使用 CloudFront 源访问控制来阻止用户直接从源访问您的内容。有关签名的 URL 和源访问控制的更多信息，请参阅使用签名 URL 和签名 Cookie 提供私有内容。

以下步骤说明如何使用第三方地理定位服务来控制对您的文件的访问。

使用第三方地理定位服务限制对 CloudFront 分配中的文件的访问

1. 获得具有地理定位服务的账户。
2. 将您的内容上传到 Amazon S3 存储桶。
3. 配置 Amazon CloudFront 和 Amazon S3 以提供私有内容。有关更多信息，请参阅 使用签名 URL 和签名 Cookie 提供私有内容。
4. 编写您的 Web 应用程序，以做到以下几点：
   - 将每个用户请求的 IP 地址发送到地理定位服务。
   - 评估地理定位服务的返回值，以确定用户是否位于您希望 CloudFront 分配您的内容的位置。
   - 如果要将内容分发到用户的位置，请为您的 CloudFront 内容生成签名的 URL。如果您不想将内容分发到该位置，请将 HTTP 状态代码 `403 (Forbidden)` 返回给用户。或者，您可配置

CloudFront 以返回自定义错误消息。有关更多信息，请参阅 the section called “为特定 HTTP 状态代码创建自定义错误页面”。

有关更多信息，请参阅您使用的地理定位服务的文档。

您可以使用 Web 服务器变量来获取正在访问您网站的用户的 IP 地址。请注意以下几点：

- 如果您的 Web 服务器未通过负载均衡器连接到 Internet，您可以使用 Web 服务器变量来获取远程 IP 地址。但是，该 IP 地址并非始终为用户的 IP 地址。它还可以是代理服务器的 IP 地址，具体取决于用户如何连接到 Internet。
- 如果您的 Web 服务器通过负载均衡器连接到 Internet，Web 服务器变量可能包含负载均衡器的 IP 地址，而不是用户的 IP 地址。在该配置中，建议您使用 `X-Forwarded-For` HTTP 标头中的最后一个 IP 地址。该标头通常包含多个 IP 地址，其中大部分是代理或负载均衡器的 IP 地址。列表中的最后一个 IP 地址是最有可能与用户的地理位置关联的 IP 地址。

如果您的 Web 服务器未连接至负载均衡器，建议您使用 Web 服务器变量，而不是 `X-Forwarded-For` 标头，以避免 IP 地址欺骗。

使用字段级加密帮助保护敏感数据

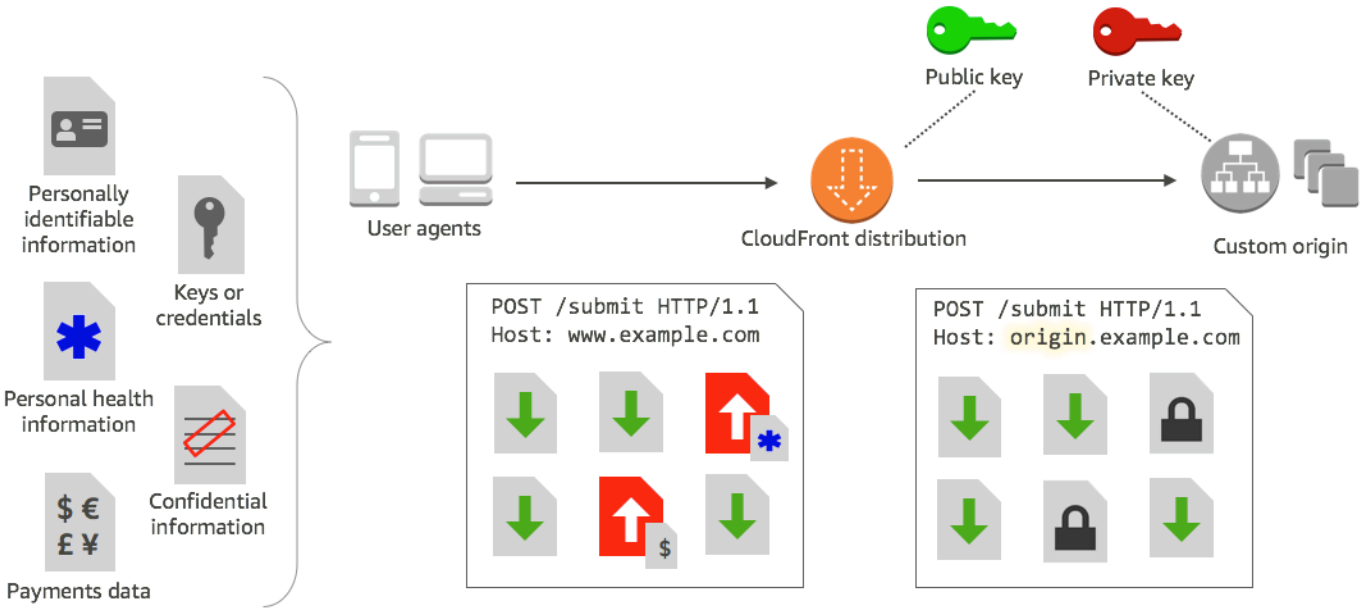
借助 Amazon CloudFront，您可以使用 HTTPS 对与源服务器的端到端连接实施保护。字段级加密增加了一个额外的安全保护层，可让您在整个系统处理过程中保护特定的数据，以便只有某些应用程序才能查看它。

借助字段级加密，您可以让您的用户安全地向您的 Web 服务器上传敏感信息。用户提供的敏感信息在靠近用户的边缘进行加密，并在整个应用程序堆栈中保持加密状态。此加密确保只有需要数据的应用程序（并且具有用于解密的凭证）能够做到这一点。

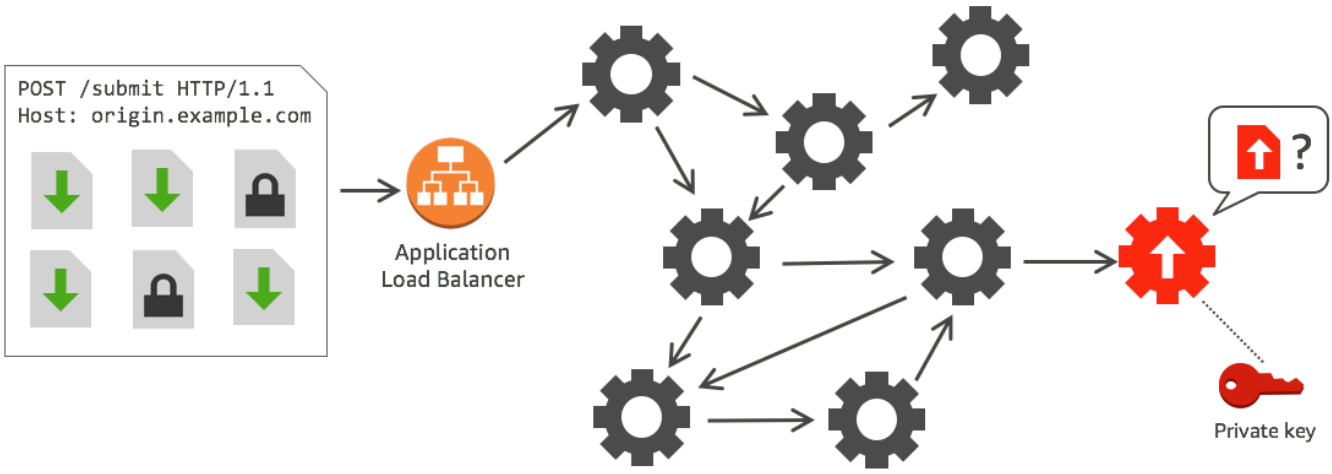
要使用字段级加密，请在配置您的 CloudFront 分配时，指定要加密的 POST 请求中的字段集以及用于对其进行加密的公有密钥。您最多可以在一个请求中加密 10 个数据字段。（您无法使用字段级加密在一个请求中加密所有数据；您必须指定要加密的各个字段。）

如果带有字段级加密的 HTTPS 请求转发到源，并且请求路由经过您的整个源应用程序或子系统，则敏感数据仍然进行加密，从而降低敏感数据泄露或意外丢失的风险。出于业务原因需要访问敏感数据的组件 (例如需要访问信用号码的支付处理系统) 可以使用适当的私有密钥来解密和访问数据。

> **Note**
>
> 要使用字段级加密，您的源必须支持分块编码。



CloudFront 字段级加密使用非对称加密，也称为公有密钥加密。您需要向 CloudFront 提供一个公有密钥，之后系统会自动加密您指定的所有敏感数据。不能使用您向 CloudFront 提供的密钥解密经过加密的值；只有您的私有密钥才能执行解密。



主题

字段级加密概览

以下步骤概述了如何设置字段级加密。有关具体步骤，请参阅设置字段级加密。

1. 获取公有密钥/私有密钥对。您必须获取和添加公有密钥，然后才能开始在 CloudFront 中设置字段级加密。
2. 创建字段级加密配置文件。您在 CloudFront 中创建的字段级加密配置文件可定义要加密的字段。
3. 创建字段级加密配置。配置指定要用于加密特定数据字段的配置文件（根据请求的内容类型或查询参数）。您还可以为不同方案选择所需的请求转发行为选项。例如，您可以设置请求 URL 中的查询参数指定的配置文件名称在 CloudFront 中不存在时的行为。
4. 链接到缓存行为。将配置链接到分配的缓存行为，用于指定 CloudFront 应加密数据的时间。

设置字段级加密

按照以下步骤操作，开始使用字段级加密。要了解有关字段级加密的配额（以前称为限制），请参阅配额。

第 1 步：创建 RSA 密钥对

要开始使用，您必须创建包含公有密钥和私有密钥的 RSA 密钥对。公有密钥使 CloudFront 能够对数据进行加密，私有密钥使源处的组件能够对已加密的字段进行解密。可以使用 OpenSSL 或其他工具创建密钥对。密钥大小必须为 2048 位。

例如，如果使用 OpenSSL，则可以使用以下命令生成一个长度为 2048 位的密钥对，并将其保存到文件 `private_key.pem` 中：

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

生成的文件同时包含公有密钥和私有密钥。要从该文件中提取公有密钥，请运行以下命令：

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

公有密钥文件 (`public_key.pem`) 包含您在以下步骤中粘贴的已编码密钥值。

第 2 步：将您的公有密钥添加到 CloudFront

获取 RSA 密钥对后，将您的公有密钥添加到 CloudFront。

将您的公有密钥添加到 CloudFront（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择公有密钥。
3. 选择添加公有密钥。
4. 对于密钥名称，请为密钥键入唯一的名称。该名称不能包含空格，且只能包含字母数字字符、下划线 (_) 和连字符 (-)。最大字符数为 128。
5. 对于 Key value (密钥值)，粘贴公有密钥的已编码密钥值，包括 `-----BEGIN PUBLIC KEY-----` 和 `-----END PUBLIC KEY-----` 行。
6. 对于注释，请添加一个可选的注释。例如，您可以包含公有密钥的到期日期。
7. 选择 Add。

您可以重复此过程中的步骤，添加更多密钥供 CloudFront 使用。

第 3 步：为字段级加密创建配置文件

在您至少将一个公有密钥添加到 CloudFront 后，创建一个配置文件，在其中告诉 CloudFront 加密哪些字段。

为字段级加密创建配置文件 (控制台)

1. 在导航窗格中，选择字段级加密。
2. 选择创建配置文件。

3. 填写以下字段：

   配置文件名称

   为配置文件键入唯一名称。该名称不能包含空格，且只能包含字母数字字符、下划线 (_) 和连字符 (-)。最大字符数为 128。

   公有密钥名称

   在下拉列表中，选择您在第 2 步中添加到 CloudFront 的公有密钥的名称。CloudFront 可使用该密钥加密您在此配置文件中指定的字段。

   提供商名称

   键入短语以帮助识别密钥，例如从其获取密钥对的提供商。在应用程序解密数据字段时，需要该信息以及私有密钥。提供商名称不能包含空格，且只能包含字母数字字符、冒号 (:)、下划线 (_) 和连字符 (-)。最大字符数为 128。

   要匹配的字段名称模式

   键入数据字段的名称，或者键入用于识别请求中希望 CloudFront 加密的数据字段名称的模式。选择 + 选项以添加您想要使用此密钥加密的所有字段。

   对于字段名称模式，您可以键入数据字段的整个名称，如 DateOfBirth，或者仅键入名称的第一部分及通配符 (*)，如 CreditCard*。除了可选的通配符 (*) 外，字段名称模式必须仅包含字母数字字符、方括号 ([和])、句点 (.)、下划线 (_) 和连字符 (-)。

   请确保您未针对不同的字段名称模式使用重叠字符。例如，如果您有一个字段名称模式 ABC*，您就不能再添加另一个字段名称模式 AB*。此外，字段名称区分大小写，并且您可以使用的字符的最大数目为 128。

   评论

   （可选）键入有关该配置文件的注释。您可以使用的最大字符数为 128。

4. 填写字段后，选择创建配置文件。
5. 如果您要添加更多配置文件，请选择添加配置文件。

第 4 步：创建配置

在您创建一个或多个字段级加密配置文件后，请创建一个配置，以便指定请求的内容类型，包括要加密的数据、要用于加密的配置文件以及用于指定希望 CloudFront 如何处理加密的其他选项。

例如，当 CloudFront 无法加密数据时，您可以指定在以下情况中，CloudFront 是阻止请求还是将请求转发到您的源：

- 当请求的内容类型未在配置中时 – 如果尚未在配置中添加某种内容类型，则可以指定 CloudFront 是应将具有该内容类型的请求转发到源而不加密数据字段，还是阻止该请求并返回错误。

> **Note**
>
> 如果您将内容类型添加到配置中，但未指定要与该类型一起使用的配置文件，则 CloudFront 始终将带有该内容类型的请求转发到源。

- 在查询参数中提供的配置文件名称未知时 – 在指定的 `fle-profile` 查询参数具有的配置文件名称对于分配不存在时，您可以指定 CloudFront 是应将请求发送到源而不加密数据字段，还是阻止该请求并返回错误。

在配置中，您还可以指定在 URL 中提供配置文件作为查询参数的操作是否覆盖您已为该查询映射到内容类型的配置文件。默认情况下，CloudFront 使用您已映射到内容类型的配置文件（如果您已指定）。这样一来，您便拥有一个默认使用的配置文件，但您可以决定要强制执行其他配置文件的某些请求。

因此，举例来说，您可以（在您的配置中）指定 **`SampleProfile`** 作为要使用的查询参数配置文件。然后，您可以使用 URL `https://d1234.cloudfront.net?fle-profile=SampleProfile` 而不是 `https://d1234.cloudfront.net` 以便 CloudFront 在该请求中使用 **`SampleProfile`** 而不是您为请求的内容类型设置的配置文件。

您可以为单个账户最多创建 10 个配置，然后将其中一个配置与该账户的任何分配的缓存行为相关联。

为字段级加密创建配置（控制台）

1. 在字段级加密页面上，选择创建配置。

   注意：如果您尚未创建至少一个配置文件，您将不会看到配置创建选项。

2. 填写以下字段以指定要使用的配置文件。（某些字段无法更改。）

   内容类型（无法更改）

   内容类型将设置为 `application/x-www-form-urlencoded` 并且无法更改。

默认配置文件 ID（可选）

在下拉列表中，选择您要映射到内容类型字段中的内容类型的配置文件。

内容格式（无法更改）

内容格式将设置为 URLencoded 并且无法更改。

3. 如果您要更改以下选项的默认 CloudFront 行为，请选择适当的复选框。

当请求的内容类型未配置时，将请求转发到原始地址

如果您要允许请求转到源，并且您尚未指定要用于请求的内容类型的配置文件，则选中该复选框。

使用提供的查询参数覆盖内容类型的配置文件

如果您要允许某个查询参数中提供的配置文件覆盖您为内容类型指定的配置文件，则选中该复选框。

4. 如果您选中该复选框以允许查询参数覆盖默认配置文件，则您必须完成以下额外配置字段。您最多可以创建五个查询参数映射供查询使用。

查询参数

键入要包含在 `fle-profile` 查询参数的 URL 中的值。对于此查询的字段级加密，此值告诉 CloudFront 使用与此查询参数关联的配置文件 ID（在下一个字段中指定）。

您可以使用的最大字符数为 128。该值不能包含空格，并且必须仅使用字母数字或以下字符：短划线 (-)、句点 (.)、下划线 (_)、星号 (*)、加号 (+)、百分号 (%)。

配置文件 ID

在下拉列表中，选择您要与为查询参数键入的值相关联的配置文件。

当查询参数中指定的配置文件不存在时，将请求转发到源

如果您要允许请求转到您的源，并且在查询参数中指定的配置文件没有在 CloudFront 中定义，则选择该复选框。

第 5 步：将配置添加到缓存行为

要使用字段级加密，需要将配置链接到分配的缓存行为，方法为添加配置 ID 作为分配的值。

> **Important**
>
> 要将字段级加密配置链接到缓存行为，必须将分配配置为始终使用 HTTPS，并接受来自查看器的 HTTP POST 和 PUT 请求。也就是说，必须满足以下条件：
>
> - 缓存行为的查看器协议策略必须设置为将 HTTP 重定向到 HTTPS 或仅 HTTPS。（在 AWS CloudFormation 或 CloudFront API 中，`ViewerProtocolPolicy` 必须设置为 `redirect-to-https` 或 `https-only`。）
> - 缓存行为的 Allowed HTTP Methods（允许的 HTTP 方法）必须设置为 GET、HEAD、OPTIONS、PUT、POST、PATCH、DELETE。（在 AWS CloudFormation 或 CloudFront API 中，`AllowedMethods` 必须设置为 `GET`、`HEAD`、`OPTIONS`、`PUT`、`POST`、`PATCH`、`DELETE`。可以按照任意顺序指定这些项。）
> - 源设置的源协议策略必须设置为匹配查看器或仅 HTTPS。（在 AWS CloudFormation 或 CloudFront API 中，`OriginProtocolPolicy` 必须设置为 `match-viewer` 或 `https-only`。）

有关更多信息，请参阅 分配设置参考。

在源端解密数据字段

CloudFront 使用 AWS Encryption SDK 来加密数据字段。在您的整个应用程序堆栈中，数据保持加密，并且只能由具有用于解密数据的凭证的应用程序进行访问。

加密后，密码文本采用 base64 编码。当您的应用程序解密源上的文本时，应用程序必须首先解码密码文本，然后使用 AWS 加密开发工具包解密数据。

以下代码示例说明了应用程序如何解密源上的数据。请注意以下几点：

- 为简化示例，此示例从工作目录中的文件内加载公有密钥和私有密钥（采用 DER 格式）。实际上，您会将私有密钥存储在安全的离线位置（如离线硬件安全模块），并将公有密钥分配给您的开发团队。
- CloudFront 在加密数据时使用特定信息，应在源上使用同一组参数进行解密。初始化主密钥时 CloudFront 使用的参数包括以下这些：
  - PROVIDER_NAME：您在创建字段级加密配置文件时指定该值。在此处使用相同的值。
  - KEY_NAME：当您将公有密钥上传到 CloudFront 时创建了其名称，然后在配置文件中指定了密钥名称。在此处使用相同的值。

- ALGORITHM：CloudFront 使用 RSA/ECB/OAEPWithSHA-256AndMGF1Padding 作为加密算法，因此，您必须使用相同的算法解密数据。
- 如果您在运行以下示例程序时使用密码文本作为输入，则解密的数据会输出到您的控制台。有关更多信息，请参阅 加密开发工具包中的 Java 代码示例AWS。

代码示例

```
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

import org.apache.commons.codec.binary.Base64;

import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoResult;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;

/**
 * Sample example of decrypting data that has been encrypted by CloudFront field-level
 encryption.
 */
public class DecryptExample {

    private static final String PRIVATE_KEY_FILENAME = "private_key.der";
    private static final String PUBLIC_KEY_FILENAME = "public_key.der";
    private static PublicKey publicKey;
    private static PrivateKey privateKey;

    // CloudFront uses the following values to encrypt data, and your origin must use
 same values to decrypt it.
    // In your own code, for PROVIDER_NAME, use the provider name that you specified
 when you created your field-level
    // encryption profile. This sample uses 'DEMO' for the value.
    private static final String PROVIDER_NAME = "DEMO";
    // In your own code, use the key name that you specified when you added your public
 key to CloudFront. This sample
```

```
    // uses 'DEMOKEY' for the key name.
    private static final String KEY_NAME = "DEMOKEY";
    // CloudFront uses this algorithm when encrypting data.
    private static final String ALGORITHM = "RSA/ECB/OAEPWithSHA-256AndMGF1Padding";

    public static void main(final String[] args) throws Exception {

        final String dataToDecrypt = args[0];

        // This sample uses files to get public and private keys.
        // In practice, you should distribute the public key and save the private key
in secure storage.
        populateKeyPair();

        System.out.println(decrypt(debase64(dataToDecrypt)));
    }

    private static String decrypt(final byte[] bytesToDecrypt) throws Exception {
        // You can decrypt the stream only by using the private key.

        // 1. Instantiate the SDK
        final AwsCrypto crypto = new AwsCrypto();

        // 2. Instantiate a JCE master key
        final JceMasterKey masterKey = JceMasterKey.getInstance(
                publicKey,
                privateKey,
                PROVIDER_NAME,
                KEY_NAME,
                ALGORITHM);

        // 3. Decrypt the data
        final CryptoResult <byte[], ? > result = crypto.decryptData(masterKey,
bytesToDecrypt);
        return new String(result.getResult());
    }

    // Function to decode base64 cipher text.
    private static byte[] debase64(final String value) {
        return Base64.decodeBase64(value.getBytes());
    }

    private static void populateKeyPair() throws Exception {
```

```
        final byte[] PublicKeyBytes =
 Files.readAllBytes(Paths.get(PUBLIC_KEY_FILENAME));
        final byte[] privateKeyBytes =
 Files.readAllBytes(Paths.get(PRIVATE_KEY_FILENAME));
        publicKey = KeyFactory.getInstance("RSA").generatePublic(new
 X509EncodedKeySpec(PublicKeyBytes));
        privateKey = KeyFactory.getInstance("RSA").generatePrivate(new
 PKCS8EncodedKeySpec(privateKeyBytes));
    }
}
```

使用 CloudFront 的点播视频和实时流视频

您可以借助任何 HTTP 源使用 CloudFront 来传输点播视频（VOD）或实时流视频。您可以在云中设置视频工作流的一种方法是将 CloudFront 与 AWS Media Services结合使用。

主题

关于流视频

您必须使用编码器打包视频内容，然后 CloudFront 才能分配内容。打包过程会创建分段，其中包含音频、视频和字幕内容。它还生成清单文件，这些文件以特定顺序描述要播放的分段以及何时播放。常见的包格式为 MPEG DASH、Apple HLS、Microsoft Smooth Streaming 和 CMAF。

VOD 流

对于 VOD 流，您的视频内容存储在服务器上，查看器能够随时观看。要创建查看器可以流式传输的资产，请使用编码器（例如 AWS Elemental MediaConvert）来对媒体文件进行格式化和打包。

在将您的视频打包成正确的格式之后，您可以将其存储在服务器或 Amazon S3 存储桶中，然后在查看器请求时使用 CloudFront 提供视频。

实时视频流

对于实时视频流，视频内容在实时活动发生时实时进行流式传输，或设置为全天候实时通道。要为广播和流式传输创建实时输出，请使用编码器（如 AWS Elemental MediaLive）压缩视频并针对查看设备设置视频格式。

将视频编码后，您可以将其存储在 AWS Elemental MediaStore 中，或使用 AWS Elemental MediaPackage 转换为不同的传输格式。使用其中任何一个源设置 CloudFront 分配来传输内容。对于创建与这些服务协同工作的分配的具体步骤和指导信息，请参阅使用 AWS Elemental MediaStore 作为源来提供视频和提供使用 AWS Elemental MediaPackage 格式化的实时视频。

Wowza 和 Unified Streaming 还提供了一些工具，您可以使用它们通过 CloudFront 流式处理视频。有关将 Wowza 与 CloudFront 结合使用的更多信息，请参阅 Wowza 文档网站上的将 Wowza Streaming

Engine 许可证引入到 CloudFront 实时 HTTP 流。有关将 Unified Streaming 与 CloudFront 结合使用来进行 VOD 流式处理的信息，请参阅 Unified Streaming 文档网站上的 CloudFront。

通过 CloudFront 提供点播视频

要使用 CloudFront 进行点播视频 (VOD) 流式传输，请使用以下服务：

- Amazon S3 以原始格式存储内容并存储转码后的视频。
- 用于将视频转码为流式传输格式的编码器（如 AWS Elemental MediaConvert）。
- CloudFront 将转码视频传输给查看器。有关 Microsoft Smooth Streaming，请参阅为 Microsoft Smooth Streaming 配置点播视频。

使用 CloudFront 创建 VOD 解决方案

1. 将您的内容上传到 Amazon S3 存储桶。要了解有关使用 Amazon S3 的更多信息，请参阅《Amazon Simple Storage Service 用户指南》https://docs.aws.amazon.com/AmazonS3/latest/dev/。
2. 通过使用 MediaConvert 作业对您的内容进行转码。作业将视频转换为查看器使用的播放器所需的格式。您还可以使用此作业来创建分辨率和比特率不同的资产。这些资产用于自适应比特率 (ABR) 流式传输，这种传输方式根据查看器的可用带宽调整查看质量。MediaConvert 将转码后的视频存储在 S3 存储桶中。
3. 使用 CloudFront 分配传输转换后的内容。查看器可以随时在任何设备上观看内容。

> **Tip**
>
> 您可以了解如何使用 AWS CloudFormation 模板部署 VOD AWS 解决方案及其所有关联组件。要查看使用模板的步骤，请参阅《AWS 上的视频点播》指南中的自动部署。

为 Microsoft Smooth Streaming 配置点播视频

您可以通过以下选项使用 CloudFront 来分配已转码为 Microsoft Smooth Streaming 格式的点播视频 (VOD) 内容：

- 指定运行 Microsoft IIS 并支持将 Smooth Streaming 作为分配源的 Web 服务器。

- 在 CloudFront 分配的缓存行为中启用 Smooth Streaming。由于您可以在分配中使用多个缓存行为，因此您可以将一个分配用于 Smooth Streaming 媒体文件以及其他内容。

> **Important**
>
> 如果指定运行 Microsoft IIS 的 Web 服务器作为源，请不 要在 CloudFront 分配的缓存行为中启用 Smooth Streaming。如果您为缓存行为启用 Smooth Streaming，则 CloudFront 无法使用 Microsoft IIS 服务器作为源。

如果您在缓存行为中启用 Smooth Streaming（即您不具有运行 Microsoft IIS 的服务器），请注意以下几点：

- 如果内容与同一缓存行为的路径模式值匹配，您仍可以使用该缓存行为分配其他内容。
- CloudFront 可以将 Amazon S3 存储桶或自定义源用于 Smooth Streaming 媒体文件。如果您为缓存行为启用 Smooth Streaming，则 CloudFront 无法使用 Microsoft IIS 服务器作为源。
- 您无法使 Smooth Streaming 格式的媒体文件失效。如果要在文件到期前更新它们，则必须将其重命名。有关更多信息，请参阅 添加、删除或替换 CloudFront 分配的内容。

有关 Smooth Streaming 客户端的信息，请参阅 Microsoft 文档网站上的 Smooth Streaming。

在 Microsoft IIS Web 服务器不是源时使用 CloudFront 分配 Smooth Streaming 文件

1. 将您的媒体文件转码为 Smooth Streaming 分片的 MP4 格式。
2. 请执行以下操作之一：
   - 如果您使用的是 CloudFront 控制台：当您创建或更新分配时，请在分配的一个或多个缓存行为中启用 Smooth Streaming。
   - 如果您使用的是 CloudFront API：针对一个或多个分配的缓存行为，将 `SmoothStreaming` 元素添加到 `DistributionConfig` 复杂类型中。
3. 将 Smooth Streaming 文件上传到您的源。
4. 创建 `clientaccesspolicy.xml` 或 `crossdomainpolicy.xml` 文件，并将其添加到可在您的分配的根目录访问的位置，例如 `https://d111111abcdef8.cloudfront.net/clientaccesspolicy.xml`。下面是一个策略示例：

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<access-policy>
<cross-domain-access>
<policy>
<allow-from http-request-headers="*">
<domain uri="*"/>
</allow-from>
<grant-to>
<resource path="/" include-subpaths="true"/>
</grant-to>
</policy>
</cross-domain-access>
</access-policy>
```

有关更多信息，请参阅 Microsoft Developer Network 网站上的让服务跨域边界可用。

5. 对于应用程序（例如媒体播放器）中的链接，请以下面的格式指定媒体文件的 URL：

   ```
   https://d111111abcdef8.cloudfront.net/video/presentation.ism/Manifest
   ```

使用 CloudFront 和 AWS Media Services 提供实时流视频

要将 AWS Media Services 与 CloudFront 配合用于向全球观众传输实时内容，请参阅以下指南。

使用 AWS Elemental MediaLive 实时编码实时视频流。要对大型视频流进行编码，MediaLive 将其压缩为可分配给查看器的较小版本（编码）。

压缩实时视频流后，您可以使用以下两个主要选项之一来准备和提供内容：

- 将内容转换为所需的格式，然后提供内容 – 如果您需要多种格式的内容，请使用 AWS Elemental MediaPackage 针对不同设备类型打包内容。当您打包内容时，您还可以实施额外的特征并添加数字版权管理 (DRM)，以防止未经授权使用您的内容。有关使用 CloudFront 提供 MediaPackage 格式化的内容的分步说明，请参阅提供使用 AWS Elemental MediaPackage 格式化的实时视频。
- 使用可扩展源存储和提供内容 – 如果 MediaLive 对内容进行编码的格式是查看器所用的所有设备所需的格式，则使用高度可扩展的源（如 AWS Elemental MediaStore）来提供内容。有关使用 CloudFront 提供存储在 MediaStore 容器中的内容的分步说明，请参阅使用 AWS Elemental MediaStore 作为源来提供视频。

使用上述选项之一设置您的源之后，您可以使用 CloudFront 将实时流视频分发给查看器。

> **Tip**
>
> 您可以了解 AWS 解决方案，该解决方案可自动部署服务以构建高度可用的实时查看体验。要查看自动部署此解决方案的步骤，请参阅实时流式自动化部署。

主题

使用 AWS Elemental MediaStore 作为源来提供视频

如果将视频存储在 AWS Elemental MediaStore 容器中，您可以创建 CloudFront 分配以提供内容。

要开始操作，您可以授予 CloudFront 对您的 MediaStore 容器的访问权限。然后，您创建一个 CloudFront 分配，并将其配置为使用 MediaStore。

从 AWS Elemental MediaStore 容器提供内容

1. 按照允许 Amazon CloudFront 访问您的 AWS Elemental MediaStore 容器中的过程进行操作，然后返回到这些步骤以创建分配。
2. 使用以下设置创建分配：

   a. 源域 – 分配到您的 MediaStore 容器的数据端点。从下拉列表中，选择用于您的实时视频的 MediaStore 容器。

   b. 源路径 – 存储您的对象的 MediaStore 容器中的文件夹结构。有关更多信息，请参阅 the section called “源路径”。

   c. 添加自定义标头 – 如果您希望 CloudFront 在将请求转发到源时添加自定义标头，请添加标头名称和值。

   d. 查看器协议策略 - 选择将 HTTP 重定向到 HTTPS。有关更多信息，请参阅 the section called “查看器协议策略”。

   e. 缓存策略和源请求策略

      - 对于缓存策略，选择创建策略，然后创建一个适合您的缓存需求和分段持续时间的缓存策略。在创建策略后，刷新缓存策略列表，然后选择您刚创建的策略。
      - 对于源请求策略，从下拉列表中选择 CORS-CustomOrigin。

对于其他设置，您可以基于其他技术要求或您企业的需求设置特定值。有关分配的所有选项的列表以及有关设置这些选项的信息，请参阅the section called “分配设置”。

3. 对于您的应用程序中的链接（例如，媒体播放器），请以您通过 CloudFront 分配其他对象所用的相同格式指定媒体文件的名称。

提供使用 AWS Elemental MediaPackage 格式化的实时视频

如果您已使用 AWS Elemental MediaPackage 设置实时流格式，您可以创建一个 CloudFront 分配，并配置缓存行为以提供实时流。以下过程假定您已使用 MediaPackage 为实时视频创建了通道并添加了端点。

要手动为 MediaPackage 创建 CloudFront 分配，请执行以下步骤：

步骤

- 步骤 1：创建和配置 CloudFront 分配
- 步骤 2：为 MediaPackage 端点的域添加源
- 步骤 3：配置所有端点的缓存行为
- 步骤 4：启用基于标头的 MediaPackage CDN 授权
- 步骤 5：使用 CloudFront 为实时流频道提供服务

步骤 1：创建和配置 CloudFront 分配

完成以下步骤，以便为您使用 MediaPackage 创建的实时视频通道设置 CloudFront 分配。

为您的实时视频通道创建分配

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择创建分配。
3. 选择用于分配的设置，包括以下内容：

   源域

   您的 MediaPackage 实时视频通道和端点所在的源。选择文本字段，然后从下拉列表中为您的实时视频选择 MediaPackage 源域。您可将一个域映射到多个源端点。

如果您已使用另一个 AWS 账户创建源域，则在此字段中键入源 URL 值。源必须是 HTTPS URL。

例如，对于像 `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8` 这样的 HLS 端点，源域是 `3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com`。

有关更多信息，请参阅 the section called “源域”。

源路径

从中提供内容的 MediaPackage 端点的路径。

未为您填写源路径字段。必须手动输入正确的源路径。

有关源路径如何工作的更多信息，请参阅the section called “源路径”。

> **Important**
>
> 需要通配符路径 * 才能路由到 CloudFront 分配中的某处。为防止与显式路径不匹配的请求路由到真实源，请为该通配符路径创建“虚拟”源。

Example ：创建“虚拟”源

在以下示例中，端点 abc123 和 def456 路由到“真实”源，但对任何其他端点的视频内容的请求会路由到没有正确子域的 `mediapackage.us-west-2.amazonaws.com`，这会导致 HTTP 404 错误。

MediaPackage 端点：

```
https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/
index.m3u8
https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/def456/
index.m3u8
```

CloudFront 源 A：

```
Domain: 3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com
```

```
Path: None
```

CloudFront 源 B：

```
Domain: mediapackage.us-west-2.amazonaws.com
Path: None
```

CloudFront 缓存行为：

```
1. Path: /out/v1/abc123/* forward to Origin A
2. Path: /out/v1/def456/* forward to Origin A
3. Path: * forward to Origin B
```

对于其他分配设置，可以基于其他技术要求或您企业的需求设置特定值。有关分配的所有选项的列表以及有关设置这些选项的信息，请参阅the section called “分配设置”。

完成选择其他分配设置后，请选择创建分配。

4. 选择刚刚创建的分配，然后选择行为。
5. 选择默认缓存行为，然后选择编辑。为您选择作为源的通道指定正确的缓存行为设置。稍后，您将添加一个或多个其他源，并为其编辑缓存行为设置。
6. 转到 CloudFront 分配页面。
7. 等到分配的上次修改时间列的值从正在部署变为某个日期和时间后，即表示 CloudFront 已创建您的分配。

步骤 2：为 MediaPackage 端点的域添加源

重复此处的步骤，将您的每个 MediaPackage 通道端点添加到您的分配，请记住需要创建一个“虚拟”源。

将其它端点作为源进行添加

1. 在 CloudFront 控制台上，选择您为通道创建的分配。
2. 选择源，然后选择创建源。
3. 对于源域，请在下拉列表中，为您的通道选择一个 MediaPackage 端点。
4. 对于其他设置，可以基于其他技术要求或您企业的需求设置值。有关更多信息，请参阅 the section called “源设置”。

5. 选择创建源。

步骤 3：配置所有端点的缓存行为

对于每个端点，您必须配置缓存行为，以添加可正确路由请求的路径模式。您指定的路径模式取决于您提供的视频格式。以下过程包括用于 Apple HLS、CMAF、DASH 和 Microsoft Smooth Streaming 格式的路径模式信息。

您通常为每个端点设置两个缓存行为：

- 父清单文件，这是您文件的索引。
- 分段，它们是视频内容的文件。

为端点创建缓存行为

1. 在 CloudFront 控制台上，选择您为通道创建的分配。
2. 选择行为，然后选择创建行为。
3. 对于路径模式，使用特定的 MediaPackage `OriginEndpoint` GUID 作为路径前缀。

   路径模式

   对于像 `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8` 这样的 HLS 端点，创建以下两个缓存行为：

   - 对于父清单和子清单，请使用 `/out/v1/abc123/*.m3u8`。
   - 对于内容段，请使用 `/out/v1/abc123/*.ts`。

   对于像 `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.m3u8` 这样的 CMAF 端点，创建以下两个缓存行为：

   - 对于父清单和子清单，请使用 `/out/v1/abc123/*.m3u8`。
   - 对于内容段，请使用 `/out/v1/abc123/*.mp4`。

   对于像 `https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.mpd` 这样的 DASH 端点，创建以下两个缓存行为：

   - 对于父清单，请使用 `/out/v1/abc123/*.mpd`。
   - 对于内容段，请使用 `/out/v1/abc123/*.mp4`。

对于像 https://3ae97e9482b0d011.mediapackage.us-west-2.amazonaws.com/out/v1/abc123/index.ism 这样的 Microsoft Smooth Streaming 端点，仅提供清单文件，因此仅创建一个缓存行为：out/v1/abc123/index.ism/*。

4. 对于每个缓存行为，指定以下设置的值：

查看器协议策略

选择将 HTTP 重定向到 HTTPS。

缓存策略和源请求策略

对于缓存策略，选择创建策略。对于新的缓存策略，请指定以下设置：

最小 TTL

设置为 5 秒或更短，以帮助防止提供过期内容。

查询字符串

对于查询字符串（在缓存键设置中），选择包括指定的查询字符串。对于允许，添加以下值，方法是键入这些值，然后选择添加项目：

- 将 m 添加为您希望 CloudFront 用作缓存基础的查询字符串参数。MediaPackage 响应始终包含标签 ?m=###，以捕获端点的修改时间。如果内容已使用此标签的不同值进行缓存，CloudFront 会请求新的清单文件，而不提供此缓存版本。
- 如果在 MediaPackage 中使用时移查看功能，请在清单文件请求（start、end 和 *.m3u8）的缓存行为中将 *.mpd 和 index.ism/* 指定为额外的查询字符串参数。通过这种方式，可根据清单文件请求中要求的时段提供该内容。有关时移查看功能和格式化内容开始和结束请求参数的更多信息，请参阅《AWS Elemental MediaPackage 用户指南》中的时移查看。
- 如果您在 MediaPackage 中使用清单筛选特征，请针对您要与清单请求（aws.manifestfilter、*.m3u8 和 *.mpd）的缓存行为结合使用的缓存策略，将 index.ism/* 指定为额外的查询字符串参数。这会将您的分配配置为将 aws.manifestfilter 查询字符串转发到您的 MediaPackage 源，需要此字符串才能使用清单筛选特征。有关更多信息，请参阅《AWS Elemental MediaPackage 用户指南》中的清单筛选。
- 如果您使用的是低延迟 HLS（LL-HLS），请指定 _HLS_msn 和 _HLS_part 作为缓存策略的附加查询字符串参数，用于清单请求的缓存行为（*.m3u8）。这会将您的分配配置为将 _HLS_msn 和 _HLS_part 查询字符串转发到 MediaPackage 源，这是 LL-HLS 阻止播放列表请求特征正常发挥作用所必需的。

5. 选择创建。
6. 创建缓存策略后，返回到缓存行为创建工作流。刷新缓存策略列表，然后选择您刚创建的策略。
7. 选择创建行为。
8. 如果端点不是 Microsoft Smooth Streaming 端点，请重复这些步骤以创建第二个缓存行为。

步骤 4：启用基于标头的 MediaPackage CDN 授权

我们建议在 MediaPackage 端点和 CloudFront 分配之间启用基于标头的 MediaPackage CDN 授权。有关更多信息，请参阅《AWS Elemental MediaPackage 用户指南》中的在 MediaPackage 中启用 CDN 授权。

步骤 5：使用 CloudFront 为实时流频道提供服务

创建分配、添加源、创建缓存行为并启用基于标头的 CDN 授权之后，您可以使用 CloudFront 提供实时流通道。CloudFront 根据您为缓存行为配置的设置，将来自查看器的请求路由到正确的 MediaPackage 端点。

对于应用程序（例如媒体播放器）中的链接，请采用 CloudFront URL 的标准格式指定媒体文件的 URL。有关更多信息，请参阅 the section called “自定义文件 URL”。

使用函数在边缘进行自定义

借助 Amazon CloudFront，您可以编写自己的代码来自定义 CloudFront 分配如何处理 HTTP 请求和响应。该代码靠近查看者（用户）运行，以最大限度地减少延迟，而且您无需管理服务器或其他基础设施。您可以编写代码以操作通过 CloudFront 的请求和响应、执行基本身份验证和授权、在边缘生成 HTTP 响应等。

您编写并附加到 CloudFront 分配的代码称为边缘函数。CloudFront 提供了两种编写和管理边缘函数的方法：

CloudFront Functions

> 您可以在 JavaScript 中编写轻量级函数，以实现大规模、注重延迟的 CDN 定制设置。CloudFront Functions 运行时环境提供亚毫秒的启动时间，可立即扩展，从而每秒处理数百万个请求，并且非常安全。CloudFront Functions 是 CloudFront 的原生功能，这意味着您可以完全在 CloudFront 中构建、测试和部署代码。

Lambda@Edge

> Lambda@Edge 是 AWS Lambda 的扩展，可为复杂的函数提供强大而灵活的计算功能，并带来更接近您的查看器的完整应用程序逻辑，并且具有高度安全性。Lambda@Edge 函数在 Node.js 或 Python 运行时环境中运行。您将函数发布到单个AWS 区域，当您关联该函数与 CloudFront 分配时，Lambda@Edge 可自动在全球复制您的代码。

如果您在 CloudFront 上运行 AWS WAF，则可以为 CloudFront Functions 和 Lambda@Edge 使用 AWS WAF 插入标头。这适用于查看器以及源请求和源响应。

主题

CloudFront Functions 与 Lambda@Edge 之间的区别

CloudFront Functions 和 Lambda@Edge 都提供了一种运行代码以响应 CloudFront 事件的方法。

CloudFront Functions 非常适合用于以下应用场景的轻量级短期运行的函数：

- 缓存键标准化 – 转变 HTTP 请求属性（标头、查询字符串、Cookie，甚至是 URL 路径）以创建合适的缓存键，这可以提高您的缓存命中率。
- 标头操作 – 在请求或响应中插入、修改或删除 HTTP 标头。例如，您可以为每个请求添加 `True-Client-IP` 标头。
- URL 重定向或重写 – 根据请求中的信息将查看器重定向到其他页面，或者将所有请求从一个路径重写到另一个路径。
- 请求授权 – 通过检查授权标头或其他请求元数据来验证哈希授权令牌，例如 JSON Web 令牌（JWT，JSON Web Token）。

要开始使用 CloudFront Functions，请参阅使用 CloudFront Functions 在边缘进行自定义。

Lambda @Edge 非常适合以下应用场景：

- 需要几毫秒或更长时间才能完成的函数
- 需要可调节 CPU 或内存的函数
- 依赖于第三方库（包括 AWS SDK，用于与其他AWS 服务服务集成）的函数
- 需要网络访问权限以使用外部服务进行处理的函数
- 需要文件系统访问权限或访问 HTTP 请求正文的函数

要开始使用 Lambda@Edge，请参阅使用 Lambda@Edge 在边缘进行自定义。

为了帮助您根据自己的应用场景来选择选项，请使用下表来了解 CloudFront Functions 与 Lambda @Edge 之间的区别。

	CloudFront Functions	Lambda@Edge
编程语言	JavaScript（兼容 ECMAScript 5.1）	Node.js 和 Python
事件来源	• 查看器请求 • 查看器响应	• 查看器请求 • 查看器响应 • 源请求 • 源响应
支持 Amazon CloudFront KeyValueStore	是	否

	CloudFront Functions	Lambda@Edge
	CloudFront KeyValueStore 仅支持 JavaScript 运行时 2.0	
扩展	每秒 10000000 个请求或更多	每个区域每秒最多 10000 个请求
函数持续时间	亚毫秒	最长 5 秒（查看器请求和查看器响应） 最长 30 秒（源请求和源响应）
最大内存 有关更多信息，请参阅 Lambda 限额。	2 MB	128 MB – 10,240 MB（10 GB）
函数代码和包含的库的最大大小	10 KB	1 MB（查看器请求和查看器响应） 50 MB（源请求和源响应）
网络访问	否	是
文件系统访问	否	是
访问请求正文	否	是
访问地理位置和设备数据	是	否（查看器请求和查看器响应） 是（源请求和源响应）
可以完全在 CloudFront 内构建和测试	是	否
函数日志记录和指标	是	是

	CloudFront Functions	Lambda@Edge
定价	提供免费套餐；按请求收费	没有免费套餐；按请求和函数持续时间收费

使用 CloudFront Functions 在边缘进行自定义

借助 CloudFront Functions，您可以在 JavaScript 中编写轻量级函数，以实现大规模、延迟敏感的 CDN 自定义。您的函数可以操作通过 CloudFront 的请求和响应、执行基本身份验证和授权、在边缘生成 HTTP 响应等。CloudFront Functions 运行时环境提供亚毫秒的启动时间，可立即扩展，从而每秒处理数百万个请求，并且非常安全。CloudFront Functions 是 CloudFront 的原生功能，这意味着您可以完全在 CloudFront 中构建、测试和部署代码。

在将 CloudFront 函数与 CloudFront 分配相关联时，CloudFront 在 CloudFront 边缘站点中截获请求和响应并将它们传递到您的函数。当发生以下事件时，您可以调用 CloudFront Functions：

- 在 CloudFront 收到查看器的请求时 (查看器请求)
- 在 CloudFront 将响应返回到查看器之前（查看器响应）

有关 CloudFront Functions 的更多信息，请参阅以下主题：

主题

- 教程：使用 CloudFront Functions 创建简单函数
- 教程：创建包含键值的 CloudFront 函数
- 编写函数代码
- 创建函数
- 测试函数
- 更新函数
- 发布函数
- 将函数与分配关联
- Amazon CloudFront KeyValueStore

教程：使用 CloudFront Functions 创建简单函数

本教程介绍如何开始使用 CloudFront Functions。您可以创建一个简单的函数，以便将查看器重定向到其他 URL，并返回自定义响应标头。

目录

先决条件

要使用 CloudFront Functions，您需要一个 CloudFront 分配。如果您还有分配，请参阅开始使用基本 CloudFront 分配。

创建函数

您可以使用 CloudFront 控制台创建一个简单函数，用于将查看器重定向到其他 URL，并返回自定义响应标头。

创建 CloudFront 函数

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择函数，然后选择创建函数。
3. 在创建函数页面上，在名称中输入函数名称，例如 *MyFunctionName*。
4. （可选）对于描述，输入函数描述，例如**Simple test function**。
5. 对于运行时，请保留默认选定的 JavaScript 版本。
6. 选择创建函数。
7. 复制以下函数代码。此函数代码将查看器重定向到其他 URL，并返回自定义响应标题。

```
function handler(event) {
    // NOTE: This example function is for a viewer request event trigger.
    // Choose viewer request for event trigger when you associate this function
 with a distribution.
    var response = {
```

```
        statusCode: 302,
        statusDescription: 'Found',
        headers: {
            'cloudfront-functions': { value: 'generated-by-CloudFront-Functions' },
            'location': { value: 'https://aws.amazon.com/cloudfront/' }
        }
    };
    return response;
}
```

8. 对于函数代码，将代码粘贴到代码编辑器中以替换默认代码。
9. 选择保存更改。
10. （可选）您可以在发布函数之前对其进行测试。本教程不介绍如何测试函数。有关更多信息，请参阅 测试函数。
11. 选择发布选项卡，然后选择发布函数。您必须 先发布该函数，然后才能将其与 CloudFront 分配相关联。
12. 接下来，您可以将函数与分配或缓存行为相关联。在 *MyFunctionName* 页面上，选择发布选项卡。

> **Warning**
>
> 在以下步骤中，选择用于测试的分配或缓存行为。不要将此演示函数与生产环境中使用的分配或缓存行为相关联。

13. 选择添加关联。
14. 在关联对话框中，选择分配和/或缓存行为。对于事件类型，请保留默认值。
15. 选择添加关联。

    关联的分配表显示了关联的分配。
16. 等待几分钟，以便相关的分配完成部署。要检查分配的状态，请在关联的分配表中选择分配，然后选择查看分配。

    当分配的状态为 Deployed（已部署）时，您即可验证函数是否正常工作。

验证函数

部署该函数后，您可以验证它是否适用于分配。

验证函数

1. 在您的 Web 浏览器中，导航到您的分配的域名（例如 https://d111111abcdef8.cloudfront.net）。

   该函数返回到浏览器的重定向，因此浏览器会自动转到 https://aws.amazon.com/cloudfront/。

2. 在命令行窗口中，您可以使用 curl 等工具，向分配的域名发送请求。

   ```
   curl -v https://d111111abcdef8.cloudfront.net/
   ```

   在响应中，您可以看到函数添加的重定向响应（302 Found）和自定义响应标头。您的响应可能类似于下例。

   Example

   ```
   curl -v https://d111111abcdef8.cloudfront.net/
   > GET / HTTP/1.1
   > Host: d111111abcdef8.cloudfront.net
   > User-Agent: curl/7.64.1
   > Accept: */*
   >
   < HTTP/1.1 302 Found
   < Server: CloudFront
   < Date: Tue, 16 Mar 2021 18:50:48 GMT
   < Content-Length: 0
   < Connection: keep-alive
   < Location: https://aws.amazon.com/cloudfront/
   < Cloudfront-Functions: generated-by-CloudFront-Functions
   < X-Cache: FunctionGeneratedResponse from cloudfront
   < Via: 1.1 3035b31bddaf14eded329f8d22cf188c.cloudfront.net (CloudFront)
   < X-Amz-Cf-Pop: PHX50-C2
   < X-Amz-Cf-Id: ULZdIz6j43uGBlXyob_JctF9x7CCbwpNniiMlmNbmwzH1YWP9FsEHg==
   ```

教程：创建包含键值的 CloudFront 函数

本教程向您演示如何在 CloudFront 函数中包含键值。键值是键值对的一部分。您可以在函数代码中包含名称（来自键值对）。函数运行时，CloudFront 会将该名称替换为相应的值。

键值对是存储在键值存储中的变量。当您在函数中使用键（而不是硬编码值）时，您的函数会更加灵活。您可以更改键的值，而无需部署代码更改。键值对也可以减小函数的大小。有关更多信息，请参阅???。

目录

先决条件

如果您不熟悉 CloudFront Functions 和键值存储，建议您按照the section called “教程：创建简单 CloudFront 函数”中的教程操作。

完成该教程后，您可以按照本教程来扩展您创建的函数。在本教程中，我们建议您首先创建键值存储。

创建键值存储

首先，创建用于您的函数的键值存储。

创建键值存储

1. 规划要包含在函数中的键值对。记下键名称。您要在某个函数中使用的键值对，必须都位于单个键值存储中。
2. 确定工作顺序。可采用以下两种方法来继续操作：
   - 创建键值存储，并将键值对添加到存储中。然后创建（或修改）函数并纳入键名称。
   - 或者，创建（或修改）函数并纳入要使用的键名称。然后创建键值存储，并添加键值对。
3. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
4. 在导航窗格中，选择函数，然后选择 KeyValueStores 选项卡。
5. 选择创建 KeyValueStore 并输入以下字段：
   - 输入存储的名称和（可选）描述。
   - 将 S3 URI 留空。在本教程中，您将手动输入键值对。

6. 选择创建。此时将显示新键值存储的详细信息页面。此页面包含一个键值对部分，该部分目前为空。

向键值存储添加键值对

接下来，手动将键值对列表添加到您之前创建的键值存储中。

向键值存储添加键值对

1. 在键值对部分，选择添加键值对。
2. 选择添加对，然后输入键和值。选中复选标记以确认您的更改，然后重复此步骤来添加更多键值对。
3. 完成后，选择保存更改，将键值对保存到键值存储中。在确认对话框中，选择完成。

现在，您拥有了一个包含一组键值对的键值存储。

将键值存储与函数相关联

现在，您已经创建了键值存储。并且您已经创建或修改了一个包含键值存储中的键名称的函数。现在可以将键值存储与该函数进行关联。您可以从函数内部创建该关联。

将键值存储与函数相关联

1. 在导航窗格中，选择函数。默认情况下，函数选项卡显示在顶部。
2. 选择函数名称，在 Associated KeyValueStore 部分中，选择 关联现有 KeyValueStore。
3. 选择键值存储并选择关联 KeyValueStore。

> **Note**
>
> 每个函数只能关联一个键值存储。

测试并发布函数代码

将键值存储与您的函数关联后，您可以测试并发布函数代码。每次修改函数代码（包括执行以下操作）时，都应始终对其进行测试：

- 将键值存储与函数相关联。
- 修改函数及其键值存储以包含新的键值对。
- 更改键值对的值。

测试并发布函数代码

1. 有关如何测试函数的信息，请参阅the section called “测试函数”。确保您选择在 DEVELOPMENT 阶段测试函数。
2. 当您准备好在 LIVE 环境中使用该函数（具有新的或修订的键值对）时，请发布该函数。

   当您发布函数时，CloudFront 会将该函数的版本从 DEVELOPMENT 阶段复制到实时阶段。该函数具有新代码，并与键值存储相关联。（无需在实时阶段再次执行此关联。）

   有关如何发布函数的信息，请参阅the section called “发布函数”。

编写函数代码

您可以使用 CloudFront Functions，在 JavaScript 中编写轻量级函数，以实现大规模、注重延迟的 CDN 定制设置。您的函数代码可以操作通过 CloudFront 的请求和响应、执行基本身份验证和授权、在边缘生成 HTTP 响应等。

如需为 CloudFront Functions 编写函数代码的帮助，请参阅以下主题。

主题

- 确定函数的用途
- CloudFront Functions 事件结构
- 适用于 CloudFront Functions 的 JavaScript 运行时系统特征
- 键值存储的帮助程序方法
- CloudFront Functions 的代码示例

确定函数的用途

在编写函数代码之前，请确定函数的用途。CloudFront Functions 中的大多数函数都具有以下用途之一。

主题

无论函数的用途是什么，handler 都是任何函数的入口点。它采用一个名为 event 的参数，该参数通过 CloudFront 传递到函数中。event 是一个 JSON 对象，它包含 HTTP 请求的表示形式（以及响应，如果您的函数修改了 HTTP 响应）。

在查看器请求事件类型中修改 HTTP 请求

您的函数可以修改 CloudFront 从查看器（客户端）收到的 HTTP 请求，然后将修改后的请求返回给 CloudFront 以继续处理。例如，您的函数代码可能会归一化缓存键或修改请求标头。

当您创建修改 HTTP 请求的函数时，请确保选择查看器请求事件类型。这意味着，在检查请求的对象是否在 CloudFront 缓存中之前，每当 CloudFront 收到来自查看器的请求时，该函数都会运行。

Example 示例

以下伪代码显示了修改 HTTP 请求的函数的结构。

```
function handler(event) {
    var request = event.request;

    // Modify the request object here.

    return request;
}
```

该函数将修改后的 request 对象返回给 CloudFront。CloudFront 继续处理返回的请求，方法是检查 CloudFront 缓存是否存在缓存命中，并在必要时将请求发送到源。

在查看器请求事件类型中生成 HTTP 响应

您的函数可以在边缘生成 HTTP 响应并直接将其返回给查看器（客户端），而无需检查缓存的响应或由 CloudFront 进一步处理。例如，您的函数代码可能会将请求重定向到新的 URL，或者检查授权并将 401 或 403 响应返回至未经授权的请求。

创建生成 HTTP 响应的函数时，请确保选择查看器请求事件类型。这意味着，每当 CloudFront 收到来自查看器的请求时，在 CloudFront 对请求进行任何进一步处理之前，该函数都会运行。

Example 示例

以下伪代码显示了生成 HTTP 响应的函数的结构。

```
function handler(event) {
    var request = event.request;

    var response = ...; // Create the response object here,
                        // using the request properties if needed.

    return response;
}
```

该函数将 `response` 对象返回给 CloudFront，CloudFront 立即将其返回给查看器，而无需检查 CloudFront 缓存或向源发送请求。

在查看器响应事件类型中修改 HTTP 响应

您的函数可以在 CloudFront 将 HTTP 响应发送到查看器（客户端）之前修改 HTTP 响应，无论响应来自 CloudFront 缓存还是源。例如，您的函数代码可能会添加或修改响应标头、状态代码和正文内容。

当您创建修改 HTTP 响应的函数时，请确保选择查看器响应事件类型。这意味着，该函数在 CloudFront 向查看器返回响应之前运行，无论响应来自 CloudFront 缓存还是源。

Example 示例

以下伪代码显示了修改 HTTP 响应的函数的结构。

```
function handler(event) {
    var request = event.request;
    var response = event.response;

    // Modify the response object here,
    // using the request properties if needed.

    return response;
}
```

该函数将修改后的 `response` 对象返回给 CloudFront，CloudFront 会立即将其返回给查看器。

相关信息

有关使用 CloudFront Functions 的更多信息，请参阅以下主题：

- 事件结构
- JavaScript 运行时功能
- 代码示例
- 边缘函数的限制

CloudFront Functions 事件结构

CloudFront Functions 在运行函数时将 `event` 对象作为输入传递给函数代码。当您测试函数时，可以创建 `event` 对象并将其传递至您的函数。创建用于测试函数的 `event` 对象时，您可以省略 `context` 对象中的 `distributionDomainName`、`distributionId` 和 `requestId` 字段。请确保标头的名称为小写字母，在生产环境中 CloudFront Functions 传递给您的函数的 `event` 对象中情况总是如此。

下面显示了此事件对象的结构概述。

```
{
    "version": "1.0",
    "context": {
        <context object>
    },
    "viewer": {
        <viewer object>
    },
    "request": {
        <request object>
    },
    "response": {
        <response object>
    }
}
```

有关更多信息，请参阅以下主题：

主题

- 版本字段
- Context 对象
- 查看器对象
- 请求对象
- 响应对象

- 状态代码和正文
- 查询字符串、标头或 Cookie 的结构
- 示例响应对象
- 示例事件对象

版本字段

`version` 字段包含一个字符串，用于指定 CloudFront Functions 事件对象的版本。当前版本为 `1.0`。

Context 对象

`context` 对象包含有关事件的上下文信息。其中包括以下字段：

`distributionDomainName`

与事件关联的分配的 CloudFront 域名（例如 d111111abcdef8.cloudfront.net）。

`distributionId`

与事件关联的分配的 ID（例如 EDFDVBD6EXAMPLE）。

`eventType`

事件类型，`viewer-request` 或 `viewer-response`。

`requestId`

唯一标识 CloudFront 请求（及其关联响应）的字符串。

查看器对象

`viewer` 对象包含一个 `ip` 字段，其值为发送请求的查看器（客户端）的 IP 地址。如果查看器请求来自 HTTP 代理或负载均衡器，则值为该代理或负载均衡器的 IP 地址。

请求对象

`request` 对象包含查看器至 CloudFront HTTP 请求的表示形式。在传递至您的函数的 `event` 对象中，`request` 对象代表 CloudFront 从查看器中接收的实际请求。

如果您的函数代码将 `request` 对象返回到 CloudFront，则它必须使用相同的结构。

`request` 对象包含以下字段：

method

请求中的 HTTP 方法。如果您的函数代码返回 request，则无法修改此字段。这是 request 对象中唯一的只读字段。

uri

所请求对象的相对路径。

> **Note**
>
> 如果您的函数修改了 uri 值，则会出现以下情况：
>
> - 新的 uri 值必须以正斜杠（/）开头。
> - 如果某个函数更改 uri 值，则它会更改查看器请求的对象。
> - 如果某个函数更改 uri 值，它不会更改请求或源请求发送到的源的缓存行为。

querystring

表示请求中的查询字符串的对象。如果请求中没有包括查询字符串，则 request 对象仍然包括空的 querystring 对象。

querystring 对象为请求中的每个查询字符串参数包含一个字段。

headers

表示请求中的 HTTP 标头的对象。如果请求包含任何 Cookie 标头，则这些标头不属于 headers 对象的一部分。Cookies 在 cookies 对象中单独表示。

headers 对象为请求中的每个标头包含一个字段。标头名称在事件对象中转换为小写，当您的函数代码添加它们时，标头名称必须为小写。当 CloudFront 函数将事件对象转换回 HTTP 请求时，标头名称中每个单词的第一个字母都会大写。单词由连字符 (-) 分隔。例如，如果您的函数代码添加了名为 example-header-name 的标头，CloudFront 会将其转换为 HTTP 请求中的 Example-Header-Name。

cookies

表示请求中的 Cookies 的对象（Cookie 标头）。

cookies 对象为请求中的每个 Cookie 包含一个字段。

有关查询字符串、标头和 Cookies 结构的更多信息，请参阅 查询字符串、标头或 Cookie 的结构。

有关示例 event 对象，请参阅 示例事件对象。

响应对象

response 对象包含 CloudFront 至查看器 HTTP 响应的表示形式。在传递至您的函数的 event 对象中，response 对象表示 CloudFront 对查看器请求的实际响应。

如果您的函数代码返回一个 response 对象，它必须使用这个相同的结构。

response 对象包含以下字段：

statusCode

响应的 HTTP 状态代码。该值是一个整数，不是字符串。

您的函数可以生成或修改 statusCode。

statusDescription

响应的 HTTP 状态描述。如果函数代码生成响应，则此字段为可选字段。

headers

表示响应中的 HTTP 标头的对象。如果响应包含任何 Set-Cookie 标头，则这些标头不属于 headers 对象的一部分。Cookies 在 cookies 对象中单独表示。

headers 对象为响应中的每个标头包含一个字段。标头名称在事件对象中转换为小写，当您的函数代码添加它们时，标头名称必须为小写。当 CloudFront 函数将事件对象转换回 HTTP 响应时，标头名称中每个单词的第一个字母都会大写。单词由连字符 (-) 分隔。例如，如果您的函数代码添加了名为 example-header-name 的标头，CloudFront 会将其转换为 HTTP 响应中的 Example-Header-Name。

cookies

表示响应中的 Cookies 的对象（Set-Cookie 标头）。

cookies 对象为响应中的每个 Cookie 包含一个字段。

body

添加 body 字段是可选的，除非您在函数中指定该字段，否则它不会出现在 response 对象中。您的函数无权访问 CloudFront 缓存或源返回的源正文。如果您未在查看器响应函数中指定 body 字段，CloudFront 缓存或源返回的源正文会返回到查看器。

如果您希望 CloudFront 将自定义正文返回查看器，请在 `data` 字段中指定正文内容，并在 `encoding` 字段中指定正文编码。您可以将编码指定为纯文本 (`"encoding": "text"`) 或 Base64 编码的内容 (`"encoding": "base64"`)。

作为快捷方式，您也可以直接在 `body` 字段 (`"body": "<specify the body content here>"`) 中指定正文内容。执行此操作时，忽略 `data` 和 `encoding` 字段。在这种情况下，CloudFront 将正文视为纯文本。

`encoding`

body 内容（`data` 字段）的编码。有效编码仅为 `text` 和 `base64`。

如果将 `encoding` 指定为 `base64`，但正文不是有效的 base64，CloudFront 将返回错误。

`data`

body 内容。

有关修改后的状态代码和正文内容的更多信息，请参阅 状态代码和正文。

有关标头和 Cookies 结构的更多信息，请参阅 查询字符串、标头或 Cookie 的结构。

有关示例 `response` 对象，请参阅 示例响应对象。

状态代码和正文

使用 CloudFront Functions，您可以更新查看器响应状态代码，将整个响应正文替换为新响应正文，或删除响应正文。在评估来自 CloudFront 缓存或源的响应的各个方面后更新查看器响应的一些常见场景包括：

- 更改状态以设置 HTTP 200 状态代码并创建要返回查看器的静态正文内容。
- 更改状态以设置将用户重定向到其他网站的 HTTP 301 或 302 状态代码。
- 决定是提供还是丢弃查看器响应的正文。

> **Note**
>
> 如果源返回 400 及以上的 HTTP 错误，则 CloudFront Functions 将无法运行。有关更多信息，请参阅 所有边缘函数的限制。

当您在处理 HTTP 响应时，CloudFront Functions 无权访问响应正文。您可以通过将正文内容设置为所需值来替换正文内容，或者通过将值设置为空来删除正文。如果您不更新函数中的正文字段，CloudFront 缓存或源返回的源正文将返回到查看器。

> **Tip**
>
> 使用 CloudFront Functions 替换正文时，请确保将相应的标头（例如 `content-encoding`、`content-type` 或 `content-length`）与新的正文内容对齐。
> 例如，如果 CloudFront 源或缓存返回 `content-encoding: gzip`，但查看器响应函数设置的正文为纯文本，则该函数还需要相应地更改 `content-encoding` 和 `content-type` 标头。

如果您的 CloudFront 函数配置为返回 400 或以上的 HTTP 错误，则您的查看器将看不到您为相同状态代码指定的自定义错误页面。

查询字符串、标头或 Cookie 的结构

查询字符串、标头和 Cookie 共享同一个结构。查询字符串可以出现在请求中。标头出现在请求和响应中。Cookie 出现在请求和响应中。

每个查询字符串、标头或 Cookie 都是父项 `querystring`、`headers` 或 `cookies` 对象内的唯一字段。字段名称是查询字符串、标头或 Cookie 的名称。每个字段都包含一个 `value` 属性，并带有查询字符串、标头或 Cookie 的值。

目录

查询字符串值或查询字符串对象

除了查询字符串对象之外，函数还可以返回查询字符串值。查询字符串值可用于按任何自定义顺序排列查询字符串参数。

Example 示例

要修改函数代码中的查询字符串，请使用如下代码。

```
var request = event.request;
request.querystring =
 'ID=42&Exp=1619740800&TTL=1440&NoValue=&querymv=val1&querymv=val2,val3';
```

标头的特殊注意事项

仅对于标头，标头名称在事件对象中转换为小写，当您的函数代码添加它们时，标头名称必须为小写。当 CloudFront 函数将事件对象转换回 HTTP 请求或响应时，标头名称中每个单词的第一个字母都会大写。单词由连字符 (-) 分隔。例如，如果您的函数代码添加了名为 `example-header-name` 的标头，CloudFront 会将其转换为 HTTP 请求或响应中的 `Example-Header-Name`。

Example 示例

请考虑 HTTP 请求中的以下 `Host` 标头。

```
Host: video.example.com
```

此标头在 `request` 对象中表示如下：

```
"headers": {
    "host": {
        "value": "video.example.com"
    }
}
```

要访问函数代码中的 `Host` 标头，请使用如下代码：

```
var request = event.request;
var host = request.headers.host.value;
```

要在函数代码中添加或修改标头，请使用如下代码（此代码添加了一个名为 `X-Custom-Header` 且带有值 `example value` 的标头）：

```
var request = event.request;
request.headers['x-custom-header'] = {value: 'example value'};
```

重复的查询字符串、标头和 Cookies（**`multiValue`** 数组）

HTTP 请求或响应可以包含多个具有相同名称的查询字符串、标头或 Cookie。在这种情况下，重复的查询字符串、标头或 Cookie 会折叠为 `request` 或 `response` 对象中的一个字段，但此字段包含名为

multiValue 的额外属性。multiValue 属性包含一个数组，其中包含每个重复查询字符串、标头或 Cookies 的值。

Example 示例

请考虑具有以下 Accept 标头的 HTTP 请求。

```
Accept: application/json
Accept: application/xml
Accept: text/html
```

这些标头在 request 对象中表示如下。

```
"headers": {
    "accept": {
        "value": "application/json",
        "multiValue": [
            {
                "value": "application/json"
            },
            {
                "value": "application/xml"
            },
            {
                "value": "text/html"
            }
        ]
    }
}
```

> **Note**
>
> 第一个标头值（在本例中为 application/json）在 value 和 multiValue 属性中重复。这样一来，您可以通过循环 multiValue 数组来访问全部值。

如果函数代码修改具有 multiValue 数组的查询字符串、标头或 Cookie，CloudFront Functions 将使用以下规则来应用更改：

1. 如果 multiValue 数组存在并进行了任何修改，则应用该修改。value 属性中的第一个元素将被忽略。

2. 否则，将应用对 value 属性的任何修改，后续值（如果存在）保持不变。

仅当 HTTP 请求或响应中包含重复的查询字符串、标头或具有相同名称的 Cookie 时，才使用 multiValue 属性，如前面的示例所示。但是，如果单个查询字符串、标头或 Cookie 中有多个值，则不使用 multiValue 属性。

Example 示例

请考虑带有一个 Accept 标头的请求，该表头中包含三个值。

```
Accept: application/json, application/xml, text/html
```

此标头在 request 对象中表示如下。

```
"headers": {
    "accept": {
        "value": "application/json, application/xml, text/html"
    }
}
```

Cookie 属性

在 HTTP 响应的 Set-Cookie 标头中，标头包含 Cookie 的名称-值对以及可选的一组用分号分隔的属性。

Example 示例

```
Set-Cookie: cookie1=val1; Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT
```

在 response 对象中，这些属性在 Cookie 字段的 attributes 属性中表示。例如，前面的 Set-Cookie 标头表示如下：

```
"cookie1": {
    "value": "val1",
    "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT"
}
```

示例响应对象

以下示例显示了一个 response 对象（查看器响应函数的输出），其中的正文已被查看器响应函数替换。

```
{
  "response": {
    "statusCode": 200,
    "statusDescription": "OK",
    "headers": {
      "date": {
        "value": "Mon, 04 Apr 2021 18:57:56 GMT"
      },
      "server": {
        "value": "gunicorn/19.9.0"
      },
      "access-control-allow-origin": {
        "value": "*"
      },
      "access-control-allow-credentials": {
        "value": "true"
      },
      "content-type": {
        "value": "text/html"
      },
      "content-length": {
        "value": "86"
      }
    },
    "cookies": {
      "ID": {
        "value": "id1234",
        "attributes": "Expires=Wed, 05 Apr 2021 07:28:00 GMT"
      },
      "Cookie1": {
        "value": "val1",
        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT",
        "multiValue": [
          {
            "value": "val1",
            "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr 2021
 07:28:00 GMT"
          },
```

```
          {
            "value": "val2",
            "attributes": "Path=/cat; Domain=example.com; Expires=Wed, 10 Jan 2021
 07:28:00 GMT"
          }
        ]
      }
    },

    // Adding the body field is optional and it will not be present in the response
 object
    // unless you specify it in your function.
    // Your function does not have access to the original body returned by the
 CloudFront
    // cache or origin.
    // If you don't specify the body field in your viewer response function, the
 original
    // body returned by the CloudFront cache or origin is returned to viewer.

     "body": {
      "encoding": "text",
      "data": "<!DOCTYPE html><html><body><p>Here is your custom content.</p></body></
html>"
    }
  }
}
```

示例事件对象

以下示例显示了一个完整的 event 对象。

> **Note**
>
> event 对象是函数的输入。您的函数只返回 request 或 response 对象，而不返回完整的 event 对象。

```
{
    "version": "1.0",
    "context": {
        "distributionDomainName": "d111111abcdef8.cloudfront.net",
        "distributionId": "EDFDVBD6EXAMPLE",
```

```
        "eventType": "viewer-response",
        "requestId": "EXAMPLEntjQpEXAMPLE_SG5Z-EXAMPLEPmPfEXAMPLEu3EqEXAMPLE=="
    },
    "viewer": {"ip": "198.51.100.11"},
    "request": {
        "method": "GET",
        "uri": "/media/index.mpd",
        "querystring": {
            "ID": {"value": "42"},
            "Exp": {"value": "1619740800"},
            "TTL": {"value": "1440"},
            "NoValue": {"value": ""},
            "querymv": {
                "value": "val1",
                "multiValue": [
                    {"value": "val1"},
                    {"value": "val2,val3"}
                ]
            }
        },
        "headers": {
            "host": {"value": "video.example.com"},
            "user-agent": {"value": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:83.0)
 Gecko/20100101 Firefox/83.0"},
            "accept": {
                "value": "application/json",
                "multiValue": [
                    {"value": "application/json"},
                    {"value": "application/xml"},
                    {"value": "text/html"}
                ]
            },
            "accept-language": {"value": "en-GB,en;q=0.5"},
            "accept-encoding": {"value": "gzip, deflate, br"},
            "origin": {"value": "https://website.example.com"},
            "referer": {"value": "https://website.example.com/videos/12345678?
action=play"},
            "cloudfront-viewer-country": {"value": "GB"}
        },
        "cookies": {
            "Cookie1": {"value": "value1"},
            "Cookie2": {"value": "value2"},
            "cookie_consent": {"value": "true"},
            "cookiemv": {
```

```
                "value": "value3",
                "multiValue": [
                    {"value": "value3"},
                    {"value": "value4"}
                ]
            }
        }
    },
    "response": {
        "statusCode": 200,
        "statusDescription": "OK",
        "headers": {
            "date": {"value": "Mon, 04 Apr 2021 18:57:56 GMT"},
            "server": {"value": "gunicorn/19.9.0"},
            "access-control-allow-origin": {"value": "*"},
            "access-control-allow-credentials": {"value": "true"},
            "content-type": {"value": "application/json"},
            "content-length": {"value": "701"}
        },
        "cookies": {
            "ID": {
                "value": "id1234",
                "attributes": "Expires=Wed, 05 Apr 2021 07:28:00 GMT"
            },
            "Cookie1": {
                "value": "val1",
                "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed, 05 Apr
2021 07:28:00 GMT",
                "multiValue": [
                    {
                        "value": "val1",
                        "attributes": "Secure; Path=/; Domain=example.com; Expires=Wed,
05 Apr 2021 07:28:00 GMT"
                    },
                    {
                        "value": "val2",
                        "attributes": "Path=/cat; Domain=example.com; Expires=Wed, 10
Jan 2021 07:28:00 GMT"
                    }
                ]
            }
        }
    }
```

```
}
```

适用于 CloudFront Functions 的 JavaScript 运行时系统特征

CloudFront Functions JavaScript 运行时系统环境符合 ECMAScript (ES) 5.1 的要求，也支持 ES 版本 6 至 9 的某些特征。

为了获得最新的功能，建议您使用 JavaScript 运行时 2.0。

与 1.0 相比，JavaScript 运行时 2.0 有以下变化：

- 可使用缓冲区模块方法
- 以下非标准字符串原型方法不可用：
  - `String.prototype.bytesFrom()`
  - `String.prototype.fromBytes()`
  - `String.prototype.fromUTF8()`
  - `String.prototype.toBytes()`
  - `String.prototype.toUTF8()`
- 加密模块包含以下更改：
  - `hash.digest()` - 如果未提供编码，则返回类型更改为 `Buffer`
  - `hmac.digest()` - 如果未提供编码，则返回类型更改为 `Buffer`
- 有关其他新功能的更多信息，请参阅适用于 CloudFront Functions 的 JavaScript 运行时系统 2.0 特征。

主题

- 适用于 CloudFront Functions 的 JavaScript 运行时 1.0 功能
- 适用于 CloudFront Functions 的 JavaScript 运行时系统 2.0 特征

适用于 CloudFront Functions 的 JavaScript 运行时 1.0 功能

CloudFront Functions JavaScript 运行时环境符合 ECMAScript (ES) 版本 5.1，也支持 ES 版本 6 至 9 的某些功能。它还提供了一些不属于 ES 规范的非标准方法。

以下主题列出了所有支持的语言功能。

主题

- 核心功能
- 原语对象
- 内置对象
- 错误类型
- 全局变量
- 内置模块
- 受限功能

核心功能

支持 ES 的以下核心功能。

类型

支持所有 ES 5.1 类型。这包括布尔值、数字、字符串、对象、数组、函数、函数构造函数和正则表达式。

运算符

支持所有 ES 5.1 运算符。

支持 ES 7 幂运算符 (**)。

语句

> **Note**
>
> 不支持 `const` 和 `let` 语句。

支持以下 ES 5.1 语句：

- `break`
- `catch`
- `continue`
- `do-while`
- `else`
- `finally`

- for
- for-in
- if
- return
- switch
- throw
- try
- var
- while
- 带标签的语句

文本

支持 ES 6 模板文本：多行字符串、表达式插值和嵌套模板。

函数

支持所有的 ES 5.1 函数功能。

支持 ES 6 箭头函数，并支持 ES 6 rest 参数语法。

Unicode

源文本和字符串文本可以包含 Unicode 编码的字符。还支持由六个字符组成的 Unicode 代码点转义序列（例如，\uXXXX）。

严格模式

默认情况下，函数在严格模式下运行，因此您无需在函数代码中添加 use strict 语句。无法对其进行更改。

原语对象

支持 ES 的以下原语对象。

对象

支持对对象使用以下 ES 5.1 方法：

- create（没有属性列表）
- defineProperties

- defineProperty
- freeze
- getOwnPropertyDescriptor
- getOwnPropertyNames
- getPrototypeOf
- hasOwnProperty
- isExtensible
- isFrozen
- prototype.isPrototypeOf
- isSealed
- keys
- preventExtensions
- prototype.propertyIsEnumerable
- seal
- prototype.toString
- prototype.valueOf

支持对对象使用以下 ES 6 方法：

- assign
- is
- prototype.setPrototypeOf

支持对对象使用以下 ES 8 方法：

- entries
- values

字符串

支持对字符串使用以下 ES 5.1 方法：

- fromCharCode
- prototype.charAt
- prototype.concat
- prototype.indexOf

- prototype.lastIndexOf
- prototype.match
- prototype.replace
- prototype.search
- prototype.slice
- prototype.split
- prototype.substr
- prototype.substring
- prototype.toLowerCase
- prototype.trim
- prototype.toUpperCase

支持对字符串使用以下 ES 6 方法：

- fromCodePoint
- prototype.codePointAt
- prototype.endsWith
- prototype.includes
- prototype.repeat
- prototype.startsWith

支持对字符串使用以下 ES 8 方法：

- prototype.padStart
- prototype.padEnd

支持对字符串使用以下 ES 9 方法：

- prototype.trimStart
- prototype.trimEnd

支持对字符串使用以下非标准方法：

- prototype.bytesFrom(*array* | *string*, *encoding*)

  从八位数组或编码字符串创建字节字符串。字符串编码选项为 hex、base64 和 base64url。
- prototype.fromBytes(*start*[, *end*])

从字节字符串创建 Unicode 字符串，其中每个字节都被替换为相应的 Unicode 代码点。

- `prototype.fromUTF8(start[, end])`

  从 UTF-8 编码的字节字符串创建 Unicode 字符串。如果编码不正确，则会返回 `null`。

- `prototype.toBytes(start[, end])`

  从 Unicode 字符串创建字节字符串。所有字符都必须在 [0,255] 范围内。如果不在该范围内，它会返回 `null`。

- `prototype.toUTF8(start[, end])`

  从 Unicode 字符串创建 UTF-8 编码的字节字符串。

数字

支持对数字使用所有 ES 5.1 方法。

支持对数字使用以下 ES 6 方法：

- `isFinite`
- `isInteger`
- `isNaN`
- `isSafeInteger`
- `parseFloat`
- `parseInt`
- `prototype.toExponential`
- `prototype.toFixed`
- `prototype.toPrecision`
- `EPSILON`
- `MAX_SAFE_INTEGER`
- `MAX_VALUE`
- `MIN_SAFE_INTEGER`
- `MIN_VALUE`
- `NEGATIVE_INFINITY`
- `NaN`
- `POSITIVE_INFINITY`

内置对象

支持 ES 的以下内置对象。

数学

支持所有 ES 5.1 数学方法。

> **Note**
>
> 在 CloudFront Functions 运行时环境中，`Math.random()` 实现使用植入有函数运行时间戳的 OpenBSD `arc4random`。

支持以下 ES 6 数学方法：

- `acosh`
- `asinh`
- `atanh`
- `cbrt`
- `clz32`
- `cosh`
- `expm1`
- `fround`
- `hypot`
- `imul`
- `log10`
- `log1p`
- `log2`
- `sign`
- `sinh`
- `tanh`
- `trunc`
- `E`
- `LN10`

- LN2
- LOG10E
- LOG2E
- PI
- SQRT1_2
- SQRT2

日期

支持所有 ES 5.1 Date 功能。

> **Note**
>
> 出于安全原因，在单个函数运行的生命周期内，Date 始终返回相同的值（函数的开始时间）。有关更多信息，请参阅 受限功能。

函数

支持 apply、bind 和 call 方法。

不支持函数构造函数。

正则表达式

支持所有 ES 5.1 正则表达式功能。正则表达式语言与 Perl 兼容。支持 ES 9 命名的捕获组。

JSON

支持所有 ES 5.1 JSON 功能，包括 parse 和 stringify。

数组

支持对数组使用以下 ES 5.1 方法：

- isArray
- prototype.concat
- prototype.every
- prototype.filter
- prototype.forEach
- prototype.indexOf

- prototype.join
- prototype.lastIndexOf
- prototype.map
- prototype.pop
- prototype.push
- prototype.reduce
- prototype.reduceRight
- prototype.reverse
- prototype.shift
- prototype.slice
- prototype.some
- prototype.sort
- prototype.splice
- prototype.unshift

支持对数组使用以下 ES 6 方法：

- of
- prototype.copyWithin
- prototype.fill
- prototype.find
- prototype.findIndex

支持对数组使用以下 ES 7 方法：

- prototype.includes

类型化数组

支持对数组使用以下 ES 6 类型化数组：

- Int8Array
- Uint8Array
- Uint8ClampedArray
- Int16Array
- Uint16Array

- Int32Array
- Uint32Array
- Float32Array
- Float64Array
- prototype.copyWithin
- prototype.fill
- prototype.join
- prototype.set
- prototype.slice
- prototype.subarray
- prototype.toString

ArrayBuffer

支持对 ArrayBuffer 使用以下方法：

- prototype.isView
- prototype.slice

Promise

支持以下承诺方法：

- reject
- resolve
- prototype.catch
- prototype.finally
- prototype.then

加密

加密模块提供标准哈希和基于哈希的消息身份验证码 (HMAC) 帮助程序。您可以使用 require('crypto') 加载模块。该模块公开了以下方法，这些方法的行为与 Node.js 对应方法完全相同：

- createHash(*algorithm*)
- hash.update(*data*)
- hash.digest([*encoding*])
- createHmac(*algorithm*, *secret key*)

- `hmac.update(data)`
- `hmac.digest([encoding])`

有关更多信息，请参阅内置模块部分中的 加密（哈希和 HMAC）。

控制台

这是调试的帮助对象。它只支持 `log()` 方法来记录日志消息。

> **Note**
>
> CloudFront Functions 不支持逗号语法，例如 `console.log('a', 'b')`。改为使用 `console.log('a' + ' ' + 'b')` 格式。

错误类型

支持以下错误对象：

- `Error`
- `EvalError`
- `InternalError`
- `MemoryError`
- `RangeError`
- `ReferenceError`
- `SyntaxError`
- `TypeError`
- `URIError`

全局变量

支持 `globalThis` 对象。

支持以下 ES 5.1 全局函数：

- `decodeURI`
- `decodeURIComponent`
- `encodeURI`

- encodeURIComponent
- isFinite
- isNaN
- parseFloat
- parseInt

支持以下全局常数：

- NaN
- Infinity
- undefined

内置模块

支持以下内置模块。

模块

- 加密（哈希和 HMAC）
- 查询字符串

加密（哈希和 HMAC）

加密模块 (crypto) 提供标准哈希和基于哈希的消息身份验证码 (HMAC) 帮助程序。您可以使用 `require('crypto')` 加载模块。该模块提供了以下方法，这些方法的行为与 Node.js 对应方法完全相同：

哈希方法

`crypto.createHash(`*`algorithm`*`)`

创建并返回哈希对象，您可以使用给定的算法：md5、sha1 或 sha256 通过它生成哈希摘要。

`hash.update(`*`data`*`)`

用给定的 data 更新哈希内容。

`hash.digest([`*`encoding`*`])`

计算使用 hash.update() 传递的所有数据的摘要。编码可以是 hex、base64 或 base64url。

HMAC 方法

`crypto.createHmac(algorithm, secret key)`

创建并返回使用给定的 algorithm 和 secret key 的 HMAC 对象。算法可以是 md5、sha1 或 sha256。

`hmac.update(data)`

用给定的 data 更新 HMAC 内容。

`hmac.digest([encoding])`

计算使用 hmac.update() 传递的所有数据的摘要。编码可以是 hex、base64 或 base64url。

查询字符串

> **Note**
>
> CloudFront Functions 事件对象自动为您解析 URL 查询字符串。这意味着，在大多数情况下您不需要使用此模块。

查询字符串模块 (querystring) 提供了解析和格式化 URL 查询字符串的方法。您可以使用 `require('querystring')` 加载模块。该模块提供了以下方法。

`querystring.escape(string)`

URL - 对给定的 string 进行编码，从而返回转义的查询字符串。该方法由 querystring.stringify() 使用，不应直接使用。

`querystring.parse(string[, separator[, equal[, options]]])`

解析查询字符串 (string) 并返回对象。

separator 参数是用于在查询字符串中分隔键和值对的子字符串。默认为 &。

equal 参数是用于在查询字符串中分隔键和值的子字符串。默认为 =。

options 参数是具有以下键的对象：

decodeURIComponent *function*

用于解码查询字符串中百分比编码字符的函数。默认为 querystring.unescape()。

maxKeys *number*

要解析的最大密钥数。默认为 1000。使用 0 的值取消键的计数限制。

默认情况下，假定查询字符串中的百分比编码字符使用 UTF-8 编码。无效的 UTF-8 序列将被替换为 U+FFFD 替换字符。

例如，对于以下查询字符串：

```
'name=value&abc=xyz&abc=123'
```

querystring.parse() 的返回值为：

```
{
name: 'value',
abc: ['xyz', '123']
}
```

querystring.decode() 是 querystring.parse() 的别名。

querystring.stringify(*object*[, *separator*[, *equal*[, *options*]]])

序列化 object 并返回查询字符串。

separator 参数是用于在查询字符串中分隔键和值对的子字符串。默认为 &。

equal 参数是用于在查询字符串中分隔键和值的子字符串。默认为 =。

options 参数是具有以下键的对象：

encodeURIComponent *function*

用于将 URL 不安全字符转换为查询字符串中的百分比编码的函数。默认为 querystring.escape()。

默认情况下，查询字符串中需要百分比编码的字符将编码为 UTF-8。要使用其他编码，请指定 encodeURIComponent 选项。

例如，在下面的代码中：

```
querystring.stringify({ name: 'value', abc: ['xyz', '123'], anotherName: '' });
```

返回值为：

```
'name=value&abc=xyz&abc=123&anotherName='
```

querystring.encode() 是 querystring.stringify() 的别名。

querystring.unescape(*string*)

对给定的 string 中的 URL 百分比编码字符进行解码，以返回未转义的查询字符串。此方法由 querystring.parse() 使用，不应直接使用。

受限功能

由于安全考虑，以下 JavaScript 语言功能或不受支持，或收到限制。

动态代码评估

不支持动态代码评估。如果尝试，eval() 和 Function 构造函数都会引发错误。例如，const sum = new Function('a', 'b', 'return a + b') 引发错误。

计时器

不支持 setTimeout()、setImmediate() 和 clearTimeout() 函数。在函数运行中没有可以推迟或生成的预置。您的函数必须同步运行才能完成。

日期和时间戳

出于安全原因，无法访问高分辨率计时器。在单个函数运行的生命周期内，查询当前时间的所有 Date 方法始终返回相同的值。返回的时间戳是函数开始运行的时间。因此，您无法度量函数中的经过时间。

文件系统访问

没有文件系统访问权限。例如，没有像 Node.js 的 fs 模块可以进行文件系统访问。

网络访问

不支持网络调用。例如，不支持 XHR、HTTP (S) 和套接字。

适用于 CloudFront Functions 的 JavaScript 运行时系统 2.0 特征

CloudFront Functions JavaScript 运行时系统环境符合 ECMAScript (ES) 5.1 的要求，也支持 ES 版本 6 至 9 的某些特征。它还提供了一些不属于 ES 规范的非标准方法。以下主题列出了此运行时系统中支持的所有特征。

主题

核心功能

支持 ES 的以下核心功能。

类型

支持所有 ES 5.1 类型。其中包括布尔值、数字、字符串、对象、数组、函数和正则表达式。

运算符

支持所有 ES 5.1 运算符。

支持 ES 7 幂运算符 (**)。

语句

支持以下 ES 5.1 语句：

- `break`
- `catch`
- `continue`
- `do-while`
- `else`
- `finally`
- `for`
- `for-in`
- `if`
- `label`

- return
- switch
- throw
- try
- var
- while

支持以下 ES 6 语句：

- async
- await
- const
- let

> **Note**
>
> async、await、const 和 let 是 JavaScript 运行时系统 2.0 中的新语句。

文本

支持 ES 6 模板文本：多行字符串、表达式插值和嵌套模板。

函数

支持所有的 ES 5.1 函数功能。

支持 ES 6 箭头函数，并支持 ES 6 rest 参数语法。

Unicode

源文本和字符串文本可以包含 Unicode 编码的字符。还支持由六个字符组成的 Unicode 代码点转义序列（例如，\uXXXX）。

严格模式

默认情况下，函数在严格模式下运行，因此您无需在函数代码中添加 use strict 语句。无法对其进行更改。

原语对象

支持 ES 的以下原语对象。

对象

支持对对象使用以下 ES 5.1 方法：

- Object.create()（没有属性列表）
- Object.defineProperties()
- Object.defineProperty()
- Object.freeze()
- Object.getOwnPropertyDescriptor()
- Object.getOwnPropertyDescriptors()
- Object.getOwnPropertyNames()
- Object.getPrototypeOf()
- Object.isExtensible()
- Object.isFrozen()
- Object.isSealed()
- Object.keys()
- Object.preventExtensions()
- Object.seal()

支持对对象使用以下 ES 6 方法：

- Object.assign()

支持对对象使用以下 ES 8 方法：

- Object.entries()
- Object.values()

支持对对象使用以下 ES 5.1 原型方法：

- Object.prototype.hasOwnProperty()
- Object.prototype.isPrototypeOf()
- Object.prototype.propertyIsEnumerable()
- Object.prototype.toString()
- Object.prototype.valueOf()

支持对对象使用以下 ES 6 原型方法：

- Object.prototype.is()
- Object.prototype.setPrototypeOf()

String

支持对字符串使用以下 ES 5.1 方法：

- String.fromCharCode()

支持对字符串使用以下 ES 6 方法：

- String.fromCodePoint()

支持对字符串使用以下 ES 5.1 原型方法：

- String.prototype.charAt()
- String.prototype.concat()
- String.prototype.indexOf()
- String.prototype.lastIndexOf()
- String.prototype.match()
- String.prototype.replace()
- String.prototype.search()
- String.prototype.slice()
- String.prototype.split()
- String.prototype.substr()
- String.prototype.substring()
- String.prototype.toLowerCase()
- String.prototype.trim()
- String.prototype.toUpperCase()

支持对字符串使用以下 ES 6 原型方法：

- String.prototype.codePointAt()
- String.prototype.endsWith()
- String.prototype.includes()
- String.prototype.repeat()
- String.prototype.startsWith()

支持对字符串使用以下 ES 8 原型方法：

- `String.prototype.padStart()`
- `String.prototype.padEnd()`

支持对字符串使用以下 ES 9 原型方法：

- `String.prototype.trimStart()`
- `String.prototype.trimEnd()`

支持对字符串使用以下 ES 12 原型方法：

- `String.prototype.replaceAll()`

> **Note**
>
> `String.prototype.replaceAll()` 是 JavaScript 运行时系统 2.0 中的新功能。

数字

支持所有 ES 5 数字。

支持对数字使用以下 ES 6 属性：

- `Number.EPSILON`
- `Number.MAX_SAFE_INTEGER`
- `Number.MIN_SAFE_INTEGER`
- `Number.MAX_VALUE`
- `Number.MIN_VALUE`
- `Number.NaN`
- `Number.NEGATIVE_INFINITY`
- `Number.POSITIVE_INFINITY`

支持对数字使用以下 ES 6 方法：

- `Number.isFinite()`
- `Number.isInteger()`
- `Number.isNaN()`
- `Number.isSafeInteger()`
- `Number.parseInt()`

- `Number.parseFloat()`

支持对数字使用以下 ES 5.1 原型方法：

- `Number.prototype.toExponential()`
- `Number.prototype.toFixed()`
- `Number.prototype.toPrecision()`

支持 ES 12 数字分隔符。

> **Note**
>
> ES 12 数字分隔符是 JavaScript 运行时系统 2.0 中的新功能。

内置对象

支持 ES 的以下内置对象。

数学

支持所有 ES 5.1 数学方法。

> **Note**
>
> 在 CloudFront Functions 运行时环境中，`Math.random()` 实现使用植入有函数运行时间戳的 OpenBSD `arc4random`。

支持以下 ES 6 数学属性：

- `Math.E`
- `Math.LN10`
- `Math.LN2`
- `Math.LOG10E`
- `Math.LOG2E`
- `Math.PI`
- `Math.SQRT1_2`
- `Math.SQRT2`

支持以下 ES 6 数学方法：

- Math.abs()
- Math.acos()
- Math.acosh()
- Math.asin()
- Math.asinh()
- Math.atan()
- Math.atan2()
- Math.atanh()
- Math.cbrt()
- Math.ceil()
- Math.clz32()
- Math.cos()
- Math.cosh()
- Math.exp()
- Math.expm1()
- Math.floor()
- Math.fround()
- Math.hypot()
- Math.imul()
- Math.log()
- Math.log1p()
- Math.log2()
- Math.log10()
- Math.max()
- Math.min()
- Math.pow()
- Math.random()
- Math.round()
- Math.sign()

- Math.sinh()
- Math.sin()
- Math.sqrt()
- Math.tan()
- Math.tanh()
- Math.trunc()

日期

支持所有 ES 5.1 Date 功能。

> **Note**
>
> 出于安全原因，在单个函数运行的生命周期内，Date 始终返回相同的值（函数的开始时间）。有关更多信息，请参阅 受限功能。

函数

支持以下 ES 5.1 原型方法：

- Function.prototype.apply()
- Function.prototype.bind()
- Function.prototype.call()

不支持函数构造函数。

正则表达式

支持所有 ES 5.1 正则表达式功能。正则表达式语言与 Perl 兼容。

支持以下 ES 5.1 原型存取器属性：

- RegExp.prototype.global
- RegExp.prototype.ignoreCase
- RegExp.protoype.multiline
- RegExp.protoype.source
- RegExp.prototype.sticky
- RegExp.prototype.flags

> **Note**
>
> `RegExp.prototype.sticky` 和 `RegExp.prototype.flags` 是 JavaScript 运行时系统 2.0 中的新功能。

支持以下 ES 5.1 原型方法：

- `RegExp.prototype.exec()`
- `RegExp.prototype.test()`
- `RegExp.prototype.toString()`
- `RegExp.prototype[@@replace]()`
- `RegExp.prototype[@@split]()`

> **Note**
>
> `RegExp.prototype[@@split]()` 是 JavaScript 运行时系统 2.0 中的新功能。

支持以下 ES 5.1 实例属性：

- `lastIndex`

支持 ES 9 命名的捕获组。

JSON

支持以下 ES 5.1 方法：

- `JSON.parse()`
- `JSON.stringify()`

数组

支持对数组使用以下 ES 5.1 方法：

- `Array.isArray()`

支持对数组使用以下 ES 6 方法：

- `Array.of()`

支持以下 ES 5.1 原型方法：

- `Array.prototype.concat()`

- Array.prototype.every()
- Array.prototype.filter()
- Array.prototype.forEach()
- Array.prototype.indexOf()
- Array.prototype.join()
- Array.prototype.lastIndexOf()
- Array.prototype.map()
- Array.prototype.pop()
- Array.prototype.push()
- Array.prototype.reduce()
- Array.prototype.reduceRight()
- Array.prototype.reverse()
- Array.prototype.shift()
- Array.prototype.slice()
- Array.prototype.some()
- Array.prototype.sort()
- Array.prototype.splice()
- Array.prototype.unshift()

支持以下 ES 6 原型方法

- Array.prototype.copyWithin()
- Array.prototype.fill()
- Array.prototype.find()
- Array.prototype.findIndex()

支持以下 ES 7 原型方法：

- Array.prototype.includes()

类型化数组

支持以下 ES 6 类型化数组构造函数：

- Float32Array
- Float64Array

- Int8Array
- Int16Array
- Int32Array
- Uint8Array
- Uint8ClampedArray
- Uint16Array
- Uint32Array

支持以下 ES 6 方法：

- TypedArray.from()
- TypedArray.of()

> **Note**
>
> TypedArray.from() 和 TypedArray.of() 是 JavaScript 运行时系统 2.0 中的新功能。

支持以下 ES 6 原型方法：

- TypedArray.prototype.copyWithin()
- TypedArray.prototype.every()
- TypedArray.prototype.fill()
- TypedArray.prototype.filter()
- TypedArray.prototype.find()
- TypedArray.prototype.findIndex()
- TypedArray.prototype.forEach()
- TypedArray.prototype.includes()
- TypedArray.prototype.indexOf()
- TypedArray.prototype.join()
- TypedArray.prototype.lastIndexOf()
- TypedArray.prototype.map()
- TypedArray.prototype.reduce()
- TypedArray.prototype.reduceRight()

- `TypedArray.prototype.reverse()`
- `TypedArray.prototype.some()`
- `TypedArray.prototype.set()`
- `TypedArray.prototype.slice()`
- `TypedArray.prototype.sort()`
- `TypedArray.prototype.subarray()`
- `TypedArray.prototype.toString()`

> **Note**
>
> `TypedArray.prototype.every()`、`TypedArray.prototype.fill()`、`TypedArray.pro`
> 和 `TypedArray.prototype.some()` 是 JavaScript 运行时系统 2.0 中的新功能。

ArrayBuffer

支持对 ArrayBuffer 使用以下 ES 6 方法：

- `isView()`

支持对 ArrayBuffer 使用以下 ES 6 原型方法：

- `ArrayBuffer.prototype.slice()`

Promise

支持对 Promise 使用 ES 6 方法：

- `Promise.all()`
- `Promise.allSettled()`
- `Promise.any()`
- `Promise.reject()`
- `Promise.resolve()`
- `Promise.race()`

> **Note**
>
> `Promise.all()`、`Promise.allSettled()`、`Promise.any()` 和
> `Promise.race()` 是 JavaScript 运行时系统 2.0 中的新功能。

支持对 Promise 使用 ES 6 原型方法：

- `Promise.prototype.catch()`
- `Promise.prototype.finally()`
- `Promise.prototype.then()`

DataView

支持以下 ES 6 原型方法：

- `DataView.prototype.getFloat32()`
- `DataView.prototype.getFloat64()`
- `DataView.prototype.getInt16()`
- `DataView.prototype.getInt32()`
- `DataView.prototype.getInt8()`
- `DataView.prototype.getUint16()`
- `DataView.prototype.getUint32()`
- `DataView.prototype.getUint8()`
- `DataView.prototype.setFloat32()`
- `DataView.prototype.setFloat64()`
- `DataView.prototype.setInt16()`
- `DataView.prototype.setInt32()`
- `DataView.prototype.setInt8()`
- `DataView.prototype.setUint16()`
- `DataView.prototype.setUint32()`
- `DataView.prototype.setUint8()`

> **Note**
>
> 所有 Dataview ES 6 原型方法都是 JavaScript 运行时系统 2.0 中的新功能。

符号

支持以下 ES 6 方法：

- `Symbol.for()`
- `Symbol.keyfor()`

> **Note**
>
> 所有符号 ES 6 方法都是 JavaScript 运行时系统 2.0 中的新功能。

文本解码器

支持以下原型方法：

- `TextDecoder.prototype.decode()`

支持以下原型存取器属性：

- `TextDecoder.prototype.encoding`
- `TextDecoder.prototype.fatal`
- `TextDecoder.prototype.ignoreBOM`

文本编码器

支持以下原型方法：

- `TextEncoder.prototype.encode()`
- `TextEncoder.prototype.encodeInto()`

错误类型

支持以下错误对象：

- `Error`
- `EvalError`
- `InternalError`
- `RangeError`
- `ReferenceError`
- `SyntaxError`
- `TypeError`
- `URIError`

全局变量

支持 `globalThis` 对象。

支持以下 ES 5.1 全局函数：

- `decodeURI()`
- `decodeURIComponent()`
- `encodeURI()`
- `encodeURIComponent()`
- `isFinite()`
- `isNaN()`
- `parseFloat()`
- `parseInt()`

支持以下 ES 6 全局函数：

- `atob()`
- `btoa()`

> **Note**
>
> `atob()` 和 `btoa()` 是 JavaScript 运行时系统 2.0 中的新功能。

支持以下全局常数：

- `NaN`
- `Infinity`
- `undefined`
- `arguments`

内置模块

支持以下内置模块。

模块

- Buffer
- 查询字符串

- 加密

Buffer

该模块提供了以下方法：

- `Buffer.alloc(size[, fill[, encoding]])`

  分配 Buffer。

  - `size`：缓冲区大小。输入一个整数。
  - `fill`：可选。输入字符串、Buffer、Uint8Array 或整数。默认值为 `0`。
  - `encoding`：可选。如果 `fill` 是字符串，请输入以下值之一：`utf8`、`hex`、`base64`、`base64url`。默认值为 `utf8`。

- `Buffer.allocUnsafe(size)`

  分配一个未初始化的 Buffer。

  - `size`：输入一个整数。

- `Buffer.byteLength(value[, encoding])`

  返回值的长度，以字节为单位。

  - `value`：字符串、Buffer、TypedArray、Dataview 或 Arraybuffer。
  - `encoding`：可选。如果 `value` 是字符串，请输入以下值之一：`utf8`、`hex`、`base64`、`base64url`。默认值为 `utf8`。

- `Buffer.compare(buffer1, buffer2)`

  比较两个 Buffer 以帮助对数组进行排序。如果它们相同，则返回 `0`，如果 `buffer1` 排在第一位，则返回 `-1`，如果 `buffer2` 排在第一位，则返回 `1`。

  - `buffer1`：输入一个 Buffer。
  - `buffer2`：输入不同的 Buffer。

- `Buffer.concat(list[, totalLength])`

  连接多个 Buffer。如果为无，则返回 `0`。最多可返回 `totalLength`。

  - `list`：输入 Buffer 的列表。请注意，这将被截断为 `totalLength`。
  - `totalLength`：可选。输入一个无符号整数。如果为空，则使用列表中的 Buffer 实例总和。

- `Buffer.from(array)`

从数组中创建 Buffer。

- array：输入从 0 到 255 的字节数组。

- Buffer.from(arrayBuffer, byteOffset[, length]))

  从 arrayBuffer 中创建视图，从偏移量 byteOffset 开始，长度为 length。

  - arrayBuffer：输入 Buffer 数组。
  - byteOffset：输入一个整数。
  - length：可选。输入一个整数。

- Buffer.from(buffer)

  创建 Buffer 的副本。

  - buffer：输入一个 Buffer。

- Buffer.from(object[, offsetOrEncoding[, length]])

  从对象创建 Buffer。如果 valueOf() 不等于对象，则返回 Buffer.from(object.valueOf(), offsetOrEncoding, length)。

  - object：输入一个对象。
  - offsetOrEncoding：可选。输入整数或编码字符串。
  - length：可选。输入一个整数。

- Buffer.from(string[, encoding])

  从字符串创建 Buffer。

  - string：输入一个字符串。
  - encoding：可选。输入以下值之一：utf8、hex、base64、base64url。默认值为 utf8。

- Buffer.isBuffer(object)

  检查 object 是否为缓冲区。返回 true 或 false。

  - object：输入一个对象。

- Buffer.isEncoding(encoding)

  检查 encoding 是否受支持。返回 true 或 false。

  - encoding：可选。输入以下值之一：utf8、hex、base64、base64url。默认值为 utf8。

该模块提供以下缓冲区原型方法：

- Buffer.prototype.compare(target[, targetStart[, targetEnd[, sourceStart[, sourceEnd]]]])

  将 Buffer 与目标进行比较。如果它们相同，则返回 0，如果 buffer 排在第一位，则返回 1，如果 target 排在第一位，则返回 -1。

  - target：输入一个 Buffer。
  - targetStart：可选。输入一个整数。默认值为 0。
  - targetEnd：可选。输入一个整数。默认为 target 长度。
  - sourceStart：可选。输入一个整数。默认值为 0。
  - sourceEnd：可选。输入一个整数。默认为 Buffer 长度。
- Buffer.prototype.copy(target[, targetStart[, sourceStart[, sourceEnd]]])

  将缓冲区复制到 target。

  - target：输入 Buffer 或 Uint8Array。
  - targetStart：可选。输入一个整数。默认值为 0。
  - sourceStart：可选。输入一个整数。默认值为 0。
  - sourceEnd：可选。输入一个整数。默认为 Buffer 长度。
- Buffer.prototype.equals(otherBuffer)

  将 Buffer 与 otherBuffer 进行比较。返回 true 或 false。

  - otherBuffer：输入一个字符串。
- Buffer.prototype.fill(value[, offset[, end][, encoding])

  使用 value 填充 Buffer。

  - value：输入字符串、Buffer 或整数。
  - offset：可选。输入一个整数。
  - end：可选。输入一个整数。
  - encoding：可选。输入以下值之一：utf8、hex、base64、base64url。默认值为 utf8。
- Buffer.prototype.includes(value[, byteOffset][, encoding])

  在 Buffer 中搜索 value。返回 true 或 false。

  - value：输入字符串、Buffer、Uint8Array 或整数。
  - byteOffset：可选。输入一个整数。
  - encoding：可选。输入以下值之一：utf8、hex、base64、base64url。默认值为 utf8。

- Buffer.prototype.indexOf(value[, byteOffset][, encoding])

  在 Buffer 中搜索第一个 value。如果找到，则返回 index；如果找不到，则返回 -1。

  - value：输入一个字符串、Buffer、Unit8Array 或 0 到 255 之间的整数。
  - byteOffset：可选。输入一个整数。
  - encoding：可选。如果 value 是字符串，则输入以下值之一：utf8、hex、base64、base64url。默认值为 utf8。
- Buffer.prototype.lastIndexOf(value[, byteOffset][, encoding])

  在 Buffer 中搜索最后一个 value。如果找到，则返回 index；如果找不到，则返回 -1。

  - value：输入一个字符串、Buffer、Unit8Array 或 0 到 255 之间的整数。
  - byteOffset：可选。输入一个整数。
  - encoding：可选。如果 value 是字符串，则输入以下值之一：utf8、hex、base64、base64url。默认值为 utf8。
- Buffer.prototype.readInt8(offset)

  从 Buffer 中按 offset 偏移量读取 Int8。

  - offset：输入一个整数。
- Buffer.prototype.readIntBE(offset, byteLength)

  从 Buffer 中按 offset 偏移量以大端序形式读取 Int。

  - offset：输入一个整数。
  - byteLength：可选。输入一个从 1 到 6 的整数。
- Buffer.prototype.readInt16BE(offset)

  从 Buffer 中按 offset 偏移量以大端序形式读取 Int16。

  - offset：输入一个整数。
- Buffer.prototype.readInt32BE(offset)

  从 Buffer 中按 offset 偏移量以大端序形式读取 Int32。

  - offset：输入一个整数。
- Buffer.prototype.readIntLE(offset, byteLength)

  从 Buffer 中按 offset 偏移量以小端序形式读取 Int。

  - offset：输入一个整数。

  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.readInt16LE(offset)

  从 Buffer 中按 offset 偏移量以小端序形式读取 Int16。

  - offset：输入一个整数。
- Buffer.prototype.readInt32LE(offset)

  从 Buffer 中按 offset 偏移量以小端序形式读取 Int32。

  - offset：输入一个整数。
- Buffer.prototype.readUInt8(offset)

  从 Buffer 中按 offset 偏移量读取 UInt8。

  - offset：输入一个整数。
- Buffer.prototype.readUIntBE(offset, byteLength)

  从 Buffer 中按 offset 偏移量以大端序形式读取 UInt。

  - offset：输入一个整数。
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.readUInt16BE(offset)

  从 Buffer 中按 offset 偏移量以大端序形式读取 UInt16。

- • offset：输入一个整数。
- Buffer.prototype.readUInt32BE(offset)

  从 Buffer 中按 offset 偏移量以大端序形式读取 UInt32。

  - offset：输入一个整数。
- Buffer.prototype.readUIntLE(offset, byteLength)

  从 Buffer 中按 offset 偏移量以小端序形式读取 UInt。

  - offset：输入一个整数。
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.readUInt16LE(offset)

  从 Buffer 中按 offset 偏移量以小端序形式读取 UInt16。

  - offset：输入一个整数。

- Buffer.prototype.readUInt32LE(offset)

  从 Buffer 中按 offset 偏移量以小端序形式读取 UInt32。

  - offset：输入一个整数。
- Buffer.prototype.readDoubleBE([offset])

  从 Buffer 中按 offset 偏移量以大端序形式读取 64 位双精度浮点数。

  - offset：可选。输入一个整数。
- Buffer.prototype.readDoubleLE([offset])

  从 Buffer 中按 offset 偏移量以小端序形式读取 64 位双精度浮点数。

  - offset：可选。输入一个整数。
- Buffer.prototype.readFloatBE([offset])

  从 Buffer 中按 offset 偏移量以大端序形式读取 32 位浮点数。

  - offset：可选。输入一个整数。
- Buffer.prototype.readFloatLE([offset])

  从 Buffer 中按 offset 偏移量以小端序形式读取 32 位浮点数。

  - offset：可选。输入一个整数。
- Buffer.prototype.subarray([start[, end]])

  返回 Buffer 的一个副本，该副本经过偏移并用新的 start 和 end 进行裁剪。

  - start：可选。输入一个整数。默认值为 0。
  - end：可选。输入一个整数。默认为缓冲区长度。
- Buffer.prototype.swap16()

  交换 Buffer 数组字节顺序，将其视为一个 16 位数字的数组。Buffer 长度必须可以被 2 整除，否则您将收到错误。
- Buffer.prototype.swap32()

  交换 Buffer 数组字节顺序，将其视为一个 32 位数字的数组。Buffer 长度必须可以被 4 整除，否则您将收到错误。
- Buffer.prototype.swap64()

交换 Buffer 数组字节顺序，将其视为一个 64 位数字的数组。Buffer 长度必须可以被 8 整除，否则您将收到错误。

- Buffer.prototype.toJSON()

  以 JSON 格式返回 Buffer。

- Buffer.prototype.toString([encoding[, start[, end]]])

  将 Buffer 从 start 到 end 转换为编码字符串。

  - encoding：可选。输入以下值之一：utf8、hex、base64 或 base64url。默认值为 utf8。
  - start：可选。输入一个整数。默认值为 0。
  - end：可选。输入一个整数。默认为缓冲区长度。

- Buffer.prototype.write(string[, offset[, length]][, encoding])

  如果有空格，则将编码的 string 写入 Buffer，如果空间不足，则写入截断的 string。

  - string：输入一个字符串。
  - offset：可选。输入一个整数。默认值为 0。
  - length：可选。输入一个整数。默认为字符串的长度。
  - encoding：可选。（可选）输入以下值之一：utf8、hex、base64 或 base64url。默认值为 utf8。

- Buffer.prototype.writeInt8(value, offset, byteLength)

  按 offset 偏移量将 byteLength 的 Int8 value 写入 Buffer。

  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。

- Buffer.prototype.writeIntBE(value, offset, byteLength)

  使用大端序按 offset 偏移量将 value 写入 Buffer。

  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。

- Buffer.prototype.writeInt16BE(value, offset, byteLength)

  使用大端序按 offset 偏移量将 value 写入 Buffer。

  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeInt32BE(value, offset, byteLength)

  使用大端序按 offset 偏移量将 value 写入 Buffer。
  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeIntLE(offset, byteLength)

  使用小端序按 offset 偏移量将 value 写入 Buffer。
  - offset：输入一个整数。
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeInt16LE(offset, byteLength)

  使用小端序按 offset 偏移量将 value 写入 Buffer。
  - offset：输入一个整数。
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeInt32LE(offset, byteLength)

  使用小端序按 offset 偏移量将 value 写入 Buffer。
  - offset：输入一个整数。
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeUInt8(value, offset, byteLength)

  按 offset 偏移量将 byteLength 的 UInt8 value 写入 Buffer。
  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeUIntBE(value, offset, byteLength)

  使用大端序按 offset 偏移量将 value 写入 Buffer。
  - value：输入一个整数。

  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeUInt16BE(value, offset, byteLength)

  使用大端序按 offset 偏移量将 value 写入 Buffer。
  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeUInt32BE(value, offset, byteLength)

  使用大端序按 offset 偏移量将 value 写入 Buffer。
  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeUIntLE(value, offset, byteLength)

  使用小端序按 offset 偏移量将 value 写入 Buffer。
  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeUInt16LE(value, offset, byteLength)

  使用小端序按 offset 偏移量将 value 写入 Buffer。
  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。
- Buffer.prototype.writeUInt32LE(value, offset, byteLength)

  使用小端序按 offset 偏移量将 value 写入 Buffer。
  - value：输入一个整数。
  - offset：输入一个整数
  - byteLength：输入一个从 1 到 6 的整数。

- Buffer.prototype.writeDoubleBE(value, [offset])

使用大端序按 offset 偏移量将 value 写入 Buffer。

- value：输入一个整数。
- offset：可选。输入一个整数。默认值为 0。

- Buffer.prototype.writeDoubleLE(value, [offset])

  使用小端序按 offset 偏移量将 value 写入 Buffer。

  - value：输入一个整数。
  - offset：可选。输入一个整数。默认值为 0。
- Buffer.prototype.writeFloatBE(value, [offset])

  使用大端序按 offset 偏移量将 value 写入 Buffer。

  - value：输入一个整数。
  - offset：可选。输入一个整数。默认值为 0。
- Buffer.prototype.writeFloatLE(value, [offset])

  使用小端序按 offset 偏移量将 value 写入 Buffer。

  - value：输入一个整数。
  - offset：可选。输入一个整数。默认值为 0。

支持以下实例方法：

- buffer[index]

  在 Buffer 中的 index 处获取和设置八位字节（字节）。

  - 获取一个从 0 到 255 的数字。或者设置一个从 0 到 255 的数字。

支持以下实例属性：

- buffer

  获取缓冲区的 ArrayBuffer 对象。
- byteOffset

  获取缓冲区的 Arraybuffer 对象的 byteOffset。
- length

获取缓冲区字节计数。

> **Note**
>
> 所有缓冲区模块方法都是 JavaScript 运行时系统 2.0 中的新功能。

查询字符串

> **Note**
>
> CloudFront Functions 事件对象自动为您解析 URL 查询字符串。这意味着，在大多数情况下您不需要使用此模块。

查询字符串模块 (querystring) 提供了解析和格式化 URL 查询字符串的方法。您可以使用 `require('querystring')` 加载模块。该模块提供了以下方法。

`querystring.escape(`*`string`*`)`

URL - 对给定的 `string` 进行编码，从而返回转义的查询字符串。该方法由 `querystring.stringify()` 使用，不应直接使用。

`querystring.parse(`*`string`*`[, `*`separator`*`[, `*`equal`*`[, `*`options`*`]]])`

解析查询字符串 (`string`) 并返回对象。

`separator` 参数是用于在查询字符串中分隔键和值对的子字符串。默认为 `&`。

`equal` 参数是用于在查询字符串中分隔键和值的子字符串。默认为 `=`。

`options` 参数是具有以下键的对象：

`decodeURIComponent` *`function`*

用于解码查询字符串中百分比编码字符的函数。默认为 `querystring.unescape()`。

`maxKeys` *`number`*

要解析的最大密钥数。默认为 `1000`。使用 `0` 的值取消键的计数限制。

默认情况下，假定查询字符串中的百分比编码字符使用 UTF-8 编码。无效的 UTF-8 序列将被替换为 U+FFFD 替换字符。

例如，对于以下查询字符串：

```
'name=value&abc=xyz&abc=123'
```

querystring.parse() 的返回值为：

```
{
name: 'value',
abc: ['xyz', '123']
}
```

querystring.decode() 是 querystring.parse() 的别名。

querystring.stringify(*object*[, *separator*[, *equal*[, *options*]]])

序列化 object 并返回查询字符串。

separator 参数是用于在查询字符串中分隔键和值对的子字符串。默认为 &。

equal 参数是用于在查询字符串中分隔键和值的子字符串。默认为 =。

options 参数是具有以下键的对象：

encodeURIComponent *function*

用于将 URL 不安全字符转换为查询字符串中的百分比编码的函数。默认为 querystring.escape()。

默认情况下，查询字符串中需要百分比编码的字符将编码为 UTF-8。要使用其他编码，请指定 encodeURIComponent 选项。

例如，在下面的代码中：

```
querystring.stringify({ name: 'value', abc: ['xyz', '123'], anotherName: '' });
```

返回值为：

```
'name=value&abc=xyz&abc=123&anotherName='
```

querystring.encode() 是 querystring.stringify() 的别名。

querystring.unescape(*string*)

对给定的 string 中的 URL 百分比编码字符进行解码，以返回未转义的查询字符串。此方法由 querystring.parse() 使用，不应直接使用。

加密

加密模块 (crypto) 提供标准哈希和基于哈希的消息身份验证码 (HMAC) 帮助程序。您可以使用 require('crypto') 加载模块。

哈希方法

crypto.createHash(*algorithm*)

创建并返回哈希对象，您可以使用给定的算法：md5、sha1 或 sha256 通过它生成哈希摘要。

hash.update(*data*)

用给定的 data 更新哈希内容。

hash.digest([*encoding*])

计算使用 hash.update() 传递的所有数据的摘要。编码可以是 hex、base64 或 base64url。

HMAC 方法

crypto.createHmac(*algorithm, secret key*)

创建并返回使用给定的 algorithm 和 secret key 的 HMAC 对象。算法可以是 md5、sha1 或 sha256。

hmac.update(*data*)

用给定的 data 更新 HMAC 内容。

hmac.digest([*encoding*])

计算使用 hmac.update() 传递的所有数据的摘要。编码可以是 hex、base64 或 base64url。

受限功能

由于安全考虑，以下 JavaScript 语言功能或不受支持，或收到限制。

动态代码评估

不支持动态代码评估。如果尝试，`eval()` 和 `Function` 构造函数都会引发错误。例如，`const sum = new Function('a', 'b', 'return a + b')` 引发错误。

计时器

不支持 `setTimeout()`、`setImmediate()` 和 `clearTimeout()` 函数。在函数运行中没有可以推迟或生成的预置。您的函数必须同步运行才能完成。

日期和时间戳

出于安全原因，无法访问高分辨率计时器。在单个函数运行的生命周期内，查询当前时间的所有 `Date` 方法始终返回相同的值。返回的时间戳是函数开始运行的时间。因此，您无法度量函数中的经过时间。

文件系统访问

没有文件系统访问权限。

网络访问

不支持网络调用。例如，不支持 XHR、HTTP (S) 和套接字。

键值存储的帮助程序方法

如果您使用 CloudFront 键值存储在您创建的函数中包含键值，则本节适用。CloudFront Functions 有一个模块，该模块提供了三种用于从键值存储中读取值的帮助程序方法。

要在函数代码中使用此模块，请确保已将相关键值存储与该函数关联。

接下来，在函数代码的第一行中添加以下语句：

```
import cf from 'cloudfront';
const kvsId = "key value store ID";
const kvsHandle = cf.kvs(kvsId);
```

您的*#### ID* 可能类似于以下内容：`a1b2c3d4-5678-90ab-cdef-EXAMPLE1`

`get()` 方法

使用此方法可返回您指定的键名称的键值。

请求

```
get("key", options);
```

- key：需要提取其值的键的名称
- options：包含一个选项 format。它可以确保函数正确地解析数据。可能的值：
  - string：（默认）UTF8 编码
  - json
  - bytes：原始二进制数据缓冲区

请求示例

```
const value = await kvsHandle.get("myFunctionKey", { format: "string"});
```

响应

响应是以使用 options 请求的格式解析为值的 promise。默认情况下，此值以字符串的形式返回。

exists() 方法

使用该方法可确定此键在键值存储中是否存在。

请求

```
exists("key");
```

请求示例

```
const exist = await kvsHandle.exists("myFunctionkey");
```

响应

响应是返回布尔值（true 或 false）的 promise。该值指定此键在键值存储中是否存在。

错误处理

当您请求的键在关联的键值存储中不存在时，get() 方法将返回错误。要管理此使用案例，可以在代码中添加 try 和 catch 块。

`meta()` 方法

使用此方法返回有关键值存储的元数据。

请求

```
meta();
```

请求示例

```
const meta = await kvsHandle.meta();
```

响应

响应是解析为具有以下属性的对象的 `promise`：

- `creationDateTime`：创建键值存储的日期和时间，采用 ISO 8601 格式。
- `lastUpdatedDateTime`：上次从源中同步键值存储的日期和时间，采用 ISO 8601 格式。该值不包括向边缘传播的时间。
- `keyCount`：上次从源同步后 KVS 中的键总数。

响应示例

```
{keyCount:3,creationDateTime:2023-11-30T23:07:55.765Z,lastUpdatedDateTime:2023-12-15T03:57:52.4
```

CloudFront Functions 的代码示例

如需开始为 CloudFront Functions 编写函数代码的帮助，请参阅以下示例。您也可在 GitHub 上的 amazon-cloudfront-functions 存储库中查找这些示例。

主题

- 向响应添加 Cache-Control 标头
- 将跨源资源共享 (CORS) 标头添加到响应
- 将跨源资源共享 (CORS) 标头添加到请求中
- 向响应添加安全标头
- 将 True-Client-IP 标头添加到请求中

- 将查看器重定向到新的 URL
- 将 index.html 添加到不包含文件名的请求 URL 中
- 验证请求中的简单令牌
- 使用 async 和 await
- 标准化查询字符串参数
- 在函数中使用键值对

向响应添加 Cache-Control 标头

以下查看器响应函数将 `Cache-Control` HTTP 标头添加到响应中。该标头使用 `max-age` 指令告诉 Web 浏览器将响应最多缓存两年（63072000 秒）。有关更多信息，请参阅 MDN Web Docs 网站上的 Cache-Control。

在 GitHub 上查看此示例。

JavaScript runtime 2.0

```
async function handler(event) {
    const response = event.response;
    const headers = response.headers;

    // Set the cache-control header
     headers['cache-control'] = {value: 'public, max-age=63072000'};

    // Return response to viewers
    return response;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var response = event.response;
    var headers = response.headers;

    // Set the cache-control header
    headers['cache-control'] = {value: 'public, max-age=63072000'};

    // Return response to viewers
    return response;
```

```
}
```

将跨源资源共享 (CORS) 标头添加到响应

以下查看器响应函数在响应未包含 `Access-Control-Allow-Origin` HTTP 标头时，将此标头添加到响应中。此标头属于跨源资源共享 (CORS) 的一部分。该标头的值 (*) 告诉 Web 浏览器允许来自任何源的代码访问此资源。有关更多信息，请参阅 MDN Web Docs 网站上的 Access-Control-Allow-Origin。

在 GitHub 上查看此示例。

JavaScript runtime 2.0

```
async function handler(event)  {
    const request = event.request;
    const response  = event.response;

    // If Access-Control-Allow-Origin CORS header is missing, add it.
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation.
    if (!response.headers['access-control-allow-origin'] &&
 request.headers['origin']) {
        response.headers['access-control-allow-origin'] = {value:
 request.headers['origin'].value};
        console.log("Access-Control-Allow-Origin was missing, adding it now.");
    }

    return response;
}
```

JavaScript runtime 1.0

```
function handler(event)  {
    var response  = event.response;
    var headers  = response.headers;

    // If Access-Control-Allow-Origin CORS header is missing, add it.
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation.
    if (!headers['access-control-allow-origin']) {
        headers['access-control-allow-origin'] = {value: "*"};
```

```
        console.log("Access-Control-Allow-Origin was missing, adding it now.");
    }

    return response;
}
```

将跨源资源共享 (CORS) 标头添加到请求中

以下查看器请求函数在请求未包含 `Origin` HTTP 标头时，将此标头添加到请求中。此标头属于跨源资源共享 (CORS) 的一部分。此示例将标头的值设置为请求的 `Host` 标头中的值。有关更多信息，请参阅 MDN Web Docs 网站上的源。

在 GitHub 上查看此示例。

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const headers = request.headers;
    const host = request.headers.host.value;

   // If origin header is missing, set it equal to the host header.
   if (!headers.origin)
       headers.origin = {value:`https://${host}`};

   return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;

   // If origin header is missing, set it equal to the host header.
   if (!headers.origin)
       headers.origin = {value:`https://${host}`};

   return request;
}
```

向响应添加安全标头

以下函数查看器响应函数将几个常见的安全相关 HTTP 标头添加到响应中。有关更多信息，请参阅 MDN Web Docs 文档中的以下页面：

- Strict-Transport-Security
- Content-Security-Policy
- X-Content-Type-Options
- X-Frame-Options
- X-XSS-Protection

在 GitHub 上查看此示例。

JavaScript runtime 2.0

```
async function handler(event) {
    const response = event.response;
    const headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation
    headers['strict-transport-security'] = { value: 'max-age=63072000;
 includeSubdomains; preload'};
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
 'self'; script-src 'self'; style-src 'self'; object-src 'none'; frame-ancestors
 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};
    headers['referrer-policy'] = {value: 'same-origin'};

    // Return the response to viewers
    return response;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var response = event.response;
```

```
    var headers = response.headers;

    // Set HTTP security headers
    // Since JavaScript doesn't allow for hyphens in variable names, we use the
 dict["key"] notation
    headers['strict-transport-security'] = { value: 'max-age=63072000;
 includeSubdomains; preload'};
    headers['content-security-policy'] = { value: "default-src 'none'; img-src
 'self'; script-src 'self'; style-src 'self'; object-src 'none'"};
    headers['x-content-type-options'] = { value: 'nosniff'};
    headers['x-frame-options'] = {value: 'DENY'};
    headers['x-xss-protection'] = {value: '1; mode=block'};

    // Return the response to viewers
    return response;
}
```

将 True-Client-IP 标头添加到请求中

以下查看器请求函数将 `True-Client-IP` HTTP 标头添加到请求中，并将查看器的 IP 地址作为标头的值。当 CloudFront 向源发送请求时，源可以确定发送请求的 CloudFront 主机的 IP 地址，但不能确定向 CloudFront 发送原始请求的查看器（客户端）的 IP 地址。此函数将添加 `True-Client-IP` 标头，以便源可以看到查看器的 IP 地址。

> **Important**
>
> 要确保 CloudFront 在源请求中包含此标头，您必须将其添加到源请求策略中允许的标头列表中。

在 GitHub 上查看此示例。

JavaScript runtime 2.0

```
async function handler(event) {
    var request = event.request;
    var clientIP = event.viewer.ip;

    //Add the true-client-ip header to the incoming request
    request.headers['true-client-ip'] = {value: clientIP};
```

```
    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var clientIP = event.viewer.ip;

    //Add the true-client-ip header to the incoming request
    request.headers['true-client-ip'] = {value: clientIP};

    return request;
}
```

将查看器重定向到新的 URL

以下查看器请求函数会生成一个响应，以便在请求来自特定国家/地区时，将查看器重定向到特定国家/地区的 URL。此函数依赖 `CloudFront-Viewer-Country` 标头的值来确定查看器所在的国家/地区。

> **Important**
>
> 要使此函数起作用，您必须将 CloudFront 配置为将 `CloudFront-Viewer-Country` 标头添加到缓存策略或源请求策略中允许的标头中，以便将其添加到传入请求中。

当查看器请求来自德国时，此示例将查看器重定向到德国特定的 URL。如果查看器请求不是来自德国，该函数将返回原始的未修改请求。

在 GitHub 上查看此示例。

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const headers = request.headers;
    const host = request.headers.host.value;
    const country = Symbol.for('DE'); // Choose a country code
    const newurl = `https://${host}/de/index.html`; // Change the redirect URL to
 your choice
```

```
    if (headers['cloudfront-viewer-country']) {
        const countryCode = Symbol.for(headers['cloudfront-viewer-country'].value);
        if (countryCode === country) {
            const response = {
                statusCode: 302,
                statusDescription: 'Found',
                headers:
                    { "location": { "value": newurl } }
                }

            return response;
        }
    }
    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var headers = request.headers;
    var host = request.headers.host.value;
    var country = 'DE' // Choose a country code
    var newurl = `https://${host}/de/index.html` // Change the redirect URL to your
 choice

    if (headers['cloudfront-viewer-country']) {
        var countryCode = headers['cloudfront-viewer-country'].value;
        if (countryCode === country) {
            var response = {
                statusCode: 302,
                statusDescription: 'Found',
                headers:
                    { "location": { "value": newurl } }
                }

            return response;
        }
    }
    return request;
}
```

有关重写和重定向的更多信息，请参阅 AWS Workshop Studio 中的使用边缘函数处理重写和重定向。

将 index.html 添加到不包含文件名的请求 URL 中

以下查看器请求函数会将 `index.html` 附加到未在 URL 中包含文件名或扩展名的请求中。此函数对于托管在 Amazon S3 存储桶中的单页应用程序或静态生成的网站非常有用。

在 GitHub 上查看此示例。

JavaScript runtime 2.0

```
async function handler(event) {
    const request = event.request;
    const uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
}
```

JavaScript runtime 1.0

```
function handler(event) {
    var request = event.request;
    var uri = request.uri;

    // Check whether the URI is missing a file name.
    if (uri.endsWith('/')) {
        request.uri += 'index.html';
    }
    // Check whether the URI is missing a file extension.
    else if (!uri.includes('.')) {
        request.uri += '/index.html';
    }

    return request;
```

```
}
```

验证请求中的简单令牌

以下查看器请求函数将验证请求的查询字符串中的 JSON Web 令牌（JWT）。如果令牌有效，该函数将原始的未修改请求返回给 CloudFront。如果令牌无效，该函数将生成错误响应。此函数使用 `crypto` 模块。有关更多信息，请参阅 内置模块。

此函数假定请求在名为 `jwt` 的查询字符串参数中包含 JWT 值。

> **Warning**
>
> 要使用此函数，您必须将密钥放在函数代码中。

在 GitHub 上查看此示例。

JavaScript runtime 2.0

```
const crypto = require('crypto');

//Response when JWT is not valid.
const response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
    }
    // check segments
    const segments = token.split('.');
    if (segments.length !== 3) {
        throw new Error('Not enough or too many segments');
    }

    // All segment should be base64
    const headerSeg = segments[0];
    const payloadSeg = segments[1];
    const signatureSeg = segments[2];
```

```
    // base64 decode and parse JSON
    const header = JSON.parse(_base64urlDecode(headerSeg));
    const payload = JSON.parse(_base64urlDecode(payloadSeg));

    if (!noVerify) {
        const signingMethod = 'sha256';
        const signingType = 'hmac';

        // Verify signature. `sign` will return base64 string.
        const signingInput = [headerSeg, payloadSeg].join('.');

        if (!_verify(signingInput, key, signingMethod, signingType, signatureSeg)) {
            throw new Error('Signature verification failed');
        }

        // Support for nbf and exp claims.
        // According to the RFC, they should be in seconds.
        if (payload.nbf && Date.now() < payload.nbf*1000) {
            throw new Error('Token not yet active');
        }

        if (payload.exp && Date.now() > payload.exp*1000) {
            throw new Error('Token expired');
        }
    }

    return payload;
}

//Function to ensure a constant time comparison to prevent
//timing side channels.
function _constantTimeEquals(a, b) {
    if (a.length != b.length) {
        return false;
    }

    var xor = 0;
    for (var i = 0; i < a.length; i++) {
    xor |= (a.charCodeAt(i) ^ b.charCodeAt(i));
    }

    return 0 === xor;
}
```

```
function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return _constantTimeEquals(signature, _sign(input, key, method));
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return Buffer.from(str, 'base64url')
}

function handler(event) {
    const request = event.request;
    //Secret key used to verify JWT token.
    //Update with your own key.
    var key = "LzdWGpAToQ1DqYuzHxE6YOqi7G3X2yvNBot9mCXfx5k";

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        console.log("Error: No JWT in the querystring");
        return response401;
    }

    const jwtToken = request.querystring.jwt.value;

    try{
        jwt_decode(jwtToken, key);
    }
    catch(e) {
        console.log(e);
        return response401;
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    console.log("Valid JWT token");
    return request;
```

```
}
```

JavaScript runtime 1.0

```
var crypto = require('crypto');

//Response when JWT is not valid.
var response401 = {
    statusCode: 401,
    statusDescription: 'Unauthorized'
};

function jwt_decode(token, key, noVerify, algorithm) {
    // check token
    if (!token) {
        throw new Error('No token supplied');
    }
    // check segments
    var segments = token.split('.');
    if (segments.length !== 3) {
        throw new Error('Not enough or too many segments');
    }

    // All segment should be base64
    var headerSeg = segments[0];
    var payloadSeg = segments[1];
    var signatureSeg = segments[2];

    // base64 decode and parse JSON
    var header = JSON.parse(_base64urlDecode(headerSeg));
    var payload = JSON.parse(_base64urlDecode(payloadSeg));

    if (!noVerify) {
        var signingMethod = 'sha256';
        var signingType = 'hmac';

        // Verify signature. `sign` will return base64 string.
        var signingInput = [headerSeg, payloadSeg].join('.');

        if (!_verify(signingInput, key, signingMethod, signingType, signatureSeg)) {
            throw new Error('Signature verification failed');
        }
```

```
        // Support for nbf and exp claims.
        // According to the RFC, they should be in seconds.
        if (payload.nbf && Date.now() < payload.nbf*1000) {
            throw new Error('Token not yet active');
        }

        if (payload.exp && Date.now() > payload.exp*1000) {
            throw new Error('Token expired');
        }
    }

    return payload;
}

function _verify(input, key, method, type, signature) {
    if(type === "hmac") {
        return (signature === _sign(input, key, method));
    }
    else {
        throw new Error('Algorithm type not recognized');
    }
}

function _sign(input, key, method) {
    return crypto.createHmac(method, key).update(input).digest('base64url');
}

function _base64urlDecode(str) {
    return String.bytesFrom(str, 'base64url')
}

function handler(event) {
    var request = event.request;

    //Secret key used to verify JWT token.
    //Update with your own key.
    var key = "LzdWGpAToQ1DqYuzHxE6YOqi7G3X2yvNBot9mCXfx5k";

    // If no JWT token, then generate HTTP redirect 401 response.
    if(!request.querystring.jwt) {
        console.log("Error: No JWT in the querystring");
        return response401;
    }
```

```
    var jwtToken = request.querystring.jwt.value;

    try{
        jwt_decode(jwtToken, key);
    }
    catch(e) {
        console.log(e);
        return response401;
    }

    //Remove the JWT from the query string if valid and return.
    delete request.querystring.jwt;
    console.log("Valid JWT token");
    return request;
}
```

使用 async 和 await

CloudFront Functions JavaScript 运行时函数 2.0 提供了 `async` 和 `await` 语法来处理 `Promise` 对象。Promise 表示延迟的结果，可以通过标记为 `async` 的函数中的 `await` 关键字进行访问。各种新的 WebCrypto 函数都使用 Promise。

有关 `Promise` 对象的更多信息，请参阅 Promise。

> **Note**
>
> 对于以下代码示例，您必须使用 JavaScript 运行时 2.0。

```
async function answer() {
    return 42;
}

// Note: async, await can be used only inside an async function.

async function handler(event) {
    // var answer_value = answer(); // returns Promise, not a 42 value
    let answer_value = await answer(); // resolves Promise, 42
    console.log("Answer"+answer_value);
    event.request.headers['answer'] = { value : ""+answer_value };
    return event.request;
```

```
}
```

以下 JavaScript 代码示例演示了如何使用 `then` 链式方法查看 Promise。您可以使用 `catch` 来查看错误。

```
async function answer() {
    return 42;
}

async function squared_answer() {
   return answer().then(value => value * value)
}
// note async, await can be used only inside async function
async function handler(event) {
    // var answer_value = answer(); // returns Promise, not a 42 value
    let answer_value = await squared_answer(); // resolves Promise, 42
    console.log("Answer"+answer_value);
    event.request.headers['answer'] = { value : ""+answer_value };
    return event.request;
}
```

标准化查询字符串参数

您可以标准化查询字符串参数以提高缓存命中率。

以下示例适用于 JavaScript 运行时 1.0 和 2.0 版。该示例演示如何在 CloudFront 将请求转发给您的源之前，通过按字母顺序排列查询字符串来提高缓存命中率。

```
function handler(event) {
     var qs=[];
     for (var key in event.request.querystring) {
         if (event.request.querystring[key].multiValue) {
             event.request.querystring[key].multiValue.forEach((mv) => {qs.push(key +
 "=" + mv.value)});
         } else {
             qs.push(key + "=" + event.request.querystring[key].value);
         }
     };

     event.request.querystring = qs.sort().join('&');
```

```
    return event.request;
}
```

在函数中使用键值对

您可以在函数中使用键值存储中的键值对。

> **Note**
>
> 对于以下代码示例，您必须使用 JavaScript 运行时 2.0。

此示例演示了一个函数，该函数使用 HTTP 请求中的 URL 内容，在键值存储中查找自定义路径。然后，CloudFront 使用该自定义路径来发出请求。此函数有助于管理作为网站一部分的多个路径。

```
import cf from 'cloudfront';

// Declare the ID of the key value store that you have associated with this function
// The import fails at runtime if the specified key value store is not associated with
 the function

const kvsId = "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111";

const kvsHandle = cf.kvs(kvsId);

async function handler(event) {
    const request = event.request;
    // Use the first segment of the pathname as key
    // For example http(s)://domain/<key>/something/else
    const pathSegments = request.uri.split('/')
    const key = pathSegments[1]
    try {
        // Replace the first path of the pathname with the value of the key
        // For example http(s)://domain/<value>/something/else
        pathSegments[1] = await kvsHandle.get(key);
        const newUri = pathSegments.join('/');
        console.log(`${request.uri} -> ${newUri}`)
        request.uri = newUri;
    } catch (err) {
        // No change to the pathname if the key is not found
        console.log(`${request.uri} | ${err}`);
    }
```

```
    return request;
}
```

创建函数

创建函数的过程分为两个阶段：

1. 将函数代码编写为 JavaScript。您可以使用来自 CloudFront 控制台的默认示例，也可以自行编写。有关更多信息，请参阅以下主题：
   - 编写函数代码
   - the section called “事件结构”
   - CloudFront Functions 的代码示例
2. 使用 CloudFront 创建函数并包括您的代码。代码位于函数内部（不是作为引用）。

Console

创建函数

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home#/functions 登录到 CloudFront 控制台，然后选择函数页面。
2. 选择创建函数。
3. 输入在 AWS 账户中唯一的函数名称，选择 JavaScript 版本，然后选择继续。此时将显示新函数的详细信息页面。

   > **Note**
   >
   > 要在函数中使用键值对，您必须选择 JavaScript 运行时 2.0。

4. 在函数代码部分，选择构建选项卡，然后输入您的函数代码。构建选项卡中包含的代码示例说明了函数代码的基本语法。
5. 选择保存更改。
6. 如果函数代码使用键值对，则必须关联键值存储。

   您可在首次创建函数时关联键值存储。或者，您也可以稍后通过更新函数来关联它。

   要立即关联键值存储，请执行以下步骤：

   - 转至关联 KeyValueStore 部分，选择关联现有 KeyValueStore。

- 选择包含函数中键值对的键值存储，然后选择关联 KeyValueStore。

CloudFront 会立即将存储与该函数关联。您无需保存此函数。

CLI

如果您使用 CLI，则通常需要首先在文件中创建函数代码，然后使用 AWS CLI 创建函数。

创建函数

1. 在文件中创建函数代码，并将其存储在计算机可以连接到的目录中。
2. 运行该命令，如示例所示。此示例使用 `fileb://` 表示法来传入文件。它还包括换行符，以使命令更具可读性。

```
aws cloudfront create-function \
    --name MaxAge \
    --function-config '{"Comment":"Max Age 2 years","Runtime":"cloudfront-
js-2.0","KeyValueStoreAssociations":{"Quantity":1,"Items":
[{"KeyValueStoreARN":"arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}]}}' \
    --function-code fileb://function-max-age-v1.js
```

> **注意**
>
> - Runtime – JavaScript 的版本。要在函数中使用键值对，您必须指定版本 2.0。
> - KeyValueStoreAssociations – 如果您的函数使用键值对，则可以在首次创建函数时关联键值存储。或者，您可以稍后使用 update-function 关联它。Quantity 始终为 1，因为每个函数只能有一个与之关联的键值存储。

该命令成功执行后，您会看到类似以下内容的输出。

```
ETag: ETVABCEXAMPLE
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years
    Runtime: cloudfront-js-2.0
    KeyValueStoreAssociations= \
      {Quantity=1, \
```

```
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-
store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
  FunctionMetadata:
    CreatedTime: '2021-04-18T20:38:56.915000+00:00'
    FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge
    LastModifiedTime: '2023-11-19T20:38:56.915000+00:00'
    Stage: DEVELOPMENT
  Name: MaxAge
  Status: UNPUBLISHED
Location: https://cloudfront.amazonaws.com/2020-05-31/function/
arn:aws:cloudfront:::function/MaxAge
```

请求中的大部分信息都是重复的。其他信息由 CloudFront 添加。

> **注意**
>
> - ETag – 每次修改键值存储时，此值都会更改。您可以使用此值和函数名称在将来引用该函数。确保您始终使用最新的 ETag。
> - FunctionARN – 您的 CloudFront 函数的 ARN。
> - 111122223333 – AWS 账户。
> - Stage – 函数的阶段（LIVE 或 DEVELOPMENT）。
> - Status – 函数的状态（PUBLISHED 或 UNPUBLISHED）。

函数在创建后，将添加到 DEVELOPMENT 阶段。我们建议您在发布函数之前对其进行测试。在您发布函数后，函数将变为 LIVE 状态。

测试函数

在将函数部署到实时阶段（生产环境）之前，您可以先对该函数进行测试以确保其按预期运行。要测试函数，您需要指定一个事件对象，该对象表示 CloudFront 分配可能在生产环境中接收的 HTTP 请求或响应。

CloudFront Functions 执行以下操作：

1. 将提供的事件对象作为输入来运行函数。
2. 返回函数的结果（修改后的事件对象）和任何函数日志或错误消息以及函数的计算利用率。有关计算利用率的更多信息，请参阅the section called “了解计算利用率”。

目录

设置事件对象

在测试函数之前，必须设置用于测试函数的事件对象。有多种选择。

选项 1：在不保存的情况下设置事件对象

您可以在 CloudFront 控制台的可视化编辑器中设置事件对象，而不将其保存。

您可以使用此事件对象从 CloudFront 控制台测试该函数，即使未保存该对象。

选项 2：在可视化编辑器中创建事件对象

您可以在 CloudFront 控制台的可视化编辑器中设置事件对象，而不将其保存。您可以为每个函数创建 10 个事件对象，以便您可以测试不同的可能输入。

以这种方式创建事件对象时，您可以使用事件对象在 CloudFront 控制台中测试该函数。您不能使用它通过 AWS API 或 SDK 来测试函数。

选项 3：使用文本编辑器创建事件对象

您可以使用文本编辑器以 JSON 格式创建事件对象。有关事件对象结构的信息，请参阅事件结构。

您可以使用此事件对象通过 CLI 测试函数。但不能使用它在 CloudFront 控制台中测试函数。

创建事件对象（选项 1 或 2）

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home#/functions 登录到 CloudFront 控制台，然后选择函数页面。

   选择您要测试的函数。
2. 在函数详细信息页面上，选择测试选项卡。
3. 对于事件类型，选择以下选项之一：

   - 如果您的函数修改了 HTTP 请求或根据请求生成了响应，请选择查看器请求。此时将显示请求部分。

- 选择查看器响应。此时将显示请求和响应部分。

4. 填写要包含在事件中的字段。您可以选择编辑 JSON 来查看原始 JSON。
5. （可选）要保存事件，请选择保存，在保存测试事件中输入名称，然后选择保存。

   您也可以选择编辑 JSON 并复制原始 JSON，然后将其保存在 CloudFront 之外您自己的文件中。

创建事件对象（选项 3）

使用文本编辑器创建事件对象。将文件存储在计算机可以连接到的目录中。

请确保遵循以下指导原则：

- 忽略 `distributionDomainName`、`distributionId` 和 `requestId` 字段。
- 标头、Cookie 和查询字符串的名称必须为小写。

以这种方式创建事件对象的一个选项是使用可视化编辑器创建示例。您可以确保示例的格式正确。然后，您可以复制原始 JSON，并将其粘贴到文本编辑器中，然后保存文件。

有关事件结构的更多信息，请参阅事件结构。

测试此函数

您可以在 CloudFront 控制台中或使用 AWS Command Line Interface（AWS CLI）测试函数。

Console

测试此函数

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home#/functions 登录到 CloudFront 控制台，然后选择函数页面。
2. 选择您要测试的函数。
3. 选择测试选项卡。
4. 确保显示了正确的事件。要从当前显示的事件切换，请在选择测试事件字段中选择另一个事件。
5. 选择测试函数。控制台显示函数的输出，包括函数日志和计算利用率。

CLI

您可以使用 aws cloudfront test-function 命令测试函数。

测试此函数

1. 打开一个命令行窗口。
2. 您必须从包含指定文件的同一个目录运行以下命令。

   此示例使用 `fileb://` 表示法来传入事件对象文件。它还包括换行符，以使命令更具可读性。

```
aws cloudfront test-function \
    --name MaxAge \
    --if-match ETVABCEXAMPLE \
    --event-object fileb://event-maxage-test01.json \
    --stage DEVELOPMENT
```

> **注意**
>
> - 您可以通过函数的名称和 ETag（在 `if-match` 参数中）来引用该函数。您可以通过事件对象在文件系统中的位置来引用该对象。
> - 阶段可以为 `DEVELOPMENT` 或 `LIVE`。

该命令成功执行后，您会看到类似以下内容的输出。

```
TestResult:
  ComputeUtilization: '21'
  FunctionErrorMessage: ''
  FunctionExecutionLogs: []
  FunctionOutput: '{"response":{"headers":{"cloudfront-functions":
{"value":"generated-by-CloudFront-Functions"},"location":{"value":"https://
aws.amazon.com/cloudfront/"}},"statusDescription":"Found","cookies":
{},"statusCode":302}}'
  FunctionSummary:
    FunctionConfig:
      Comment: MaxAge function
      Runtime: cloudfront-js-2.0
      KeyValueStoreAssociations= \
      {Quantity=1, \
```

```
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-
store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
    FunctionMetadata:
      CreatedTime: '2021-04-18T20:38:56.915000+00:00'
      FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge
      LastModifiedTime: '2023-17-20T10:38:57.057000+00:00'
      Stage: DEVELOPMENT
    Name: MaxAge
    Status: UNPUBLISHED
```

注意

- FunctionExecutionLogs 包含函数在 console.log() 语句中写入的日志行列表（如果有）。
- ComputeUtilization 包含有关运行函数的信息。请参阅 the section called “了解计算利用率”。
- FunctionOutput 包含函数返回的事件对象。

了解计算利用率

计算利用率是函数运行所花费的时间占最大允许时间的百分比。例如，值为 35 表示函数在最大允许时间的 35% 内完成。

如果函数持续超过最大允许时间，CloudFront 将限制该函数。以下列表说明了函数根据计算利用率值受到限制的可能性。

计算利用率值：

- 1 – 50 – 该函数远低于最大允许时间，应在不受限制的情况下运行。
- 51 – 70 – 函数正在接近最大允许时间。考虑优化函数代码。
- 71 – 100 – 函数非常接近或超过最大允许时间。如果您将此函数与分配关联，CloudFront 可能会限制此函数。

更新函数

您可以随时更新函数。仅对处于 DEVELOPMENT 阶段的函数版本进行更改。要将更新从 DEVELOPMENT 阶段复制到 LIVE，您必须发布函数。

您可以在 CloudFront 控制台中或使用 AWS Command Line Interface（AWS CLI）更新函数的代码。

Console

更新函数代码

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home#/functions 登录到 CloudFront 控制台，然后选择函数页面。

   选择要更新的函数。
2. 选择编辑并进行所需的更改：

   - 根据需要在详细信息部分中更新任意字段。
   - 更改或移除关联的键值存储。有关键值存储的更多信息，请参阅the section called “使用 CloudFront KeyValueStore”。
   - 更改函数代码。选择构建选项卡，进行更改，然后选择保存更改来保存对代码的更改。

CLI

更新函数代码

1. 打开一个命令行窗口。
2. 运行以下命令。

   此示例使用 `fileb://` 表示法来传入文件。它还包括换行符，以使命令更具可读性。

```
aws cloudfront update-function \
    --name MaxAge \
    --function-config '{"Comment":"Max Age 2 years","Runtime":"cloudfront-
js-2.0","KeyValueStoreAssociations":{"Quantity":1,"Items":
[{"KeyValueStoreARN":"arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"}]}}' \
```

```
    --function-code fileb://function-max-age-v1.js \
    --if-match ETVABCEXAMPLE
```

> **注意**
>
> - 您可以通过函数的名称及其 ETag（在 `if-match` 参数中）来标识函数。确保您使用当前的 ETag。您可以使用描述操作来获取它。
> - 即使您不想对 `function-code` 进行更改，也必须包含它。
> - 务必注意 `function-config`。您应传递您想要在配置中保留的所有内容。具体而言，应按如下方式处理键值存储：
>   - 要保留现有的键值存储关联（如果有），请指定现有存储的名称。
>   - 要更改关联，请指定新键值存储的名称。
>   - 要移除关联，请忽略 `KeyValueStoreAssociations` 参数。

该命令成功执行后，您会看到类似以下内容的输出。

```
ETag: ETVXYZEXAMPLE
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years \
    Runtime: cloudfront-js-2.0 \
    KeyValueStoreAssociations= \
      {Quantity=1, \
      Items=[{KeyValueStoreARN='arn:aws:cloudfront::111122223333:key-value-
store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'}]} \
  FunctionMetadata: \
    CreatedTime: '2021-04-18T20:38:56.915000+00:00' \
    FunctionARN: arn:aws:cloudfront::111122223333:function/MaxAge \
    LastModifiedTime: '2023-12-19T23:41:15.389000+00:00' \
    Stage: DEVELOPMENT \
  Name: MaxAge \
  Status: UNPUBLISHED
```

请求中的大部分信息都是重复的。其他信息由 CloudFront 添加。

> **注意**
>
> - ETag – 每次修改键值存储时，此值都会更改。
> - FunctionARN – 您的 CloudFront 函数的 ARN。
> - Stage – 函数的阶段（LIVE 或 DEVELOPMENT）。
> - Status – 函数的状态（PUBLISHED 或 UNPUBLISHED）。

发布函数

当您发布函数时，这会将函数从 DEVELOPMENT 阶段复制到 LIVE 阶段。

如果缓存行为没有与函数关联，则发布函数使您能够将其与缓存行为相关联。您只能将缓存行为与处于 LIVE 阶段中的函数相关联。

> **Important**
>
> - 在您发布之前，我们建议您测试函数。
> - 发布函数之后，在分配完成部署时，与该函数关联的所有缓存行为会立即自动开始使用新发布的副本。

您可以在 CloudFront 控制台中或使用 AWS CLI 发布函数。

Console

发布函数

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home#/functions 登录到 CloudFront 控制台，然后选择函数页面。
2. 选择要更新的函数。
3. 选择发布选项卡，然后选择发布。如果您的函数已附加到一个或多个缓存行为，则选择发布并更新。
4. （可选）要查看与函数关联的分配，请选择关联的 CloudFront 分配以展开该部分。

成功后，页面顶部会显示一个横幅，说明####已成功发布。您还可以选择构建选项卡，然后选择实时以查看函数代码的实时版本。

CLI

发布函数

1. 打开一个命令行窗口。
2. 运行以下 aws cloudfront publish-function 命令：在示例中，提供换行符以使示例更具可读性。

```
aws cloudfront publish-function \
    --name MaxAge \
    --if-match ETVXYZEXAMPLE
```

该命令成功执行后，您会看到类似以下内容的输出。

```
FunctionSummary:
  FunctionConfig:
    Comment: Max Age 2 years
    Runtime: cloudfront-js-2.0
  FunctionMetadata:
    CreatedTime: '2021-04-18T21:24:21.314000+00:00'
    FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
    LastModifiedTime: '2023-12-19T23:41:15.389000+00:00'
    Stage: LIVE
  Name: MaxAge
  Status: UNASSOCIATED
```

将函数与分配关联

要将 CloudFront Functions 中的函数与分配结合使用，请将该函数与分配中的一个或多个缓存行为关联。您可以将函数与多个分配中的多个缓存行为相关联。

将函数与缓存行为关联时，必须选择事件类型。事件类型决定 CloudFront Functions 何时运行函数。您可以从以下事件类型中选择：

- 查看器请求 – 当 CloudFront 收到来自查看器的请求时，该函数将运行。
- 查看器请求 – 该函数在 CloudFront 向查看器返回响应之前运行。

您不能将面向源的事件类型（源请求和源响应）与 CloudFront Functions 结合使用。此时应该使用 Lambda@Edge。有关更多信息，请参阅 可以触发 Lambda@Edge 函数的 CloudFront 事件。

> **Note**
>
> 在关联函数之前，必须将其发布到 LIVE 阶段。

您可以在 CloudFront 控制台中或使用 AWS Command Line Interface（AWS CLI）将函数与分配相关联。

Console

您可以使用 CloudFront 控制台将函数与现有 CloudFront 分配中的现有缓存行为相关联。有关创建分配的更多信息，请参阅the section called “创建分配”。

将函数与现有缓存行为关联

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home#/functions 登录到 CloudFront 控制台，然后选择函数页面。
2. 选择要关联的函数。
3. 在函数页面上，选择发布选项卡。
4. 选择发布函数。
5. 选择添加关联。在出现的对话框中，选择分配、事件类型和/或缓存行为。

   对于事件类型，请选择您希望此函数在何时运行：

   - 查看器请求 – 在 CloudFront 每次收到请求时运行函数。
   - 查看器响应 – 在 CloudFront 每次返回响应时运行函数。
6. 要保存配置，请选择添加关联。

CloudFront 将分配与函数关联。等待几分钟，以便相关的分配完成部署。您可以在函数详细信息页面上选择查看分配来查看进度。

CLI

您可以将函数与以下任意一项相关联：

- 现有缓存行为

- 现有分配中的新缓存行为
- 新分配中的新缓存行为

以下程序演示如何将函数与现有缓存行为关联。

将函数与现有缓存行为关联

1. 打开一个命令行窗口。
2. 输入以下命令，以针对您要将其缓存行为与函数相关联的分配，保存分配的配置。此命令将分配配置保存到名为 `dist-config.yaml` 的文件中。要使用此命令，请执行以下操作：

   - 将 *`DistributionID`* 替换为分配的 ID。
   - 在一行上运行该命令。在示例中，提供换行符以使示例更具可读性。

   ```
   aws cloudfront get-distribution-config \
       --id DistributionID \
       --output yaml > dist-config.yaml
   ```

   命令成功后，AWS CLI 不返回任何输出。
3. 打开您创建的名为 `dist-config.yaml` 的文件。编辑该文件以进行以下更改。

   a. 将 `ETag` 字段重命名为 `IfMatch`，但不更改字段的值。

   b. 在缓存行为中，找到名为 `FunctionAssociations` 的对象。更新此对象以添加函数关联。函数关联的 YAML 语法如以下示例所示。

      - 下面的示例显示查看器请求事件类型（触发器）。要使用查看器响应事件类型，请将 `viewer-request` 替换为 `viewer-response`。
      - 将 *`arn:aws:cloudfront::111122223333:function/ExampleFunction`* 替换为您将其与此缓存行为关联的函数的 Amazon 资源名称 (ARN)。要获取函数 ARN，您可以使用 aws cloudfront list-functions 命令。

      ```
      FunctionAssociations:
        Items:
          - EventType: viewer-request
            FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
        Quantity: 1
      ```

c. 进行这些更改后，保存文件。

4. 使用以下命令更新分配，以添加函数关联。要使用此命令，请执行以下操作：

   - 将 *DistributionID* 替换为分配的 ID。
   - 在一行上运行该命令。在示例中，提供换行符以使示例更具可读性。

```
aws cloudfront update-distribution \
    --id DistributionID \
    --cli-input-yaml file://dist-config.yaml
```

命令成功后，您会看到类似以下内容的输出，该输出描述了刚通过函数关联更新的分配。为了便于阅读，下面的示例输出被截断了。

```
Distribution:
  ARN: arn:aws:cloudfront::111122223333:distribution/EBEDLT3BGRBBW
  ... truncated ...
  DistributionConfig:
    ... truncated ...
    DefaultCacheBehavior:
      ... truncated ...
      FunctionAssociations:
        Items:
        - EventType: viewer-request
          FunctionARN: arn:aws:cloudfront::111122223333:function/ExampleFunction
        Quantity: 1
      ... truncated ...
  DomainName: d111111abcdef8.cloudfront.net
  Id: EDFDVBD6EXAMPLE
  LastModifiedTime: '2021-04-19T22:39:09.158000+00:00'
  Status: InProgress
ETag: E2VJGGQEG1JT8S
```

分配的 `Status` 会在重新部署分配时更改为 `InProgress`。一旦新的分配配置到达 CloudFront 边缘站点，该边缘站点就会开始使用关联的函数。当分配完全部署后，`Status` 会更改回到 `Deployed`，这表明关联的 CloudFront 函数在全球所有的 CloudFront 边缘站点都已启用。这通常需要花费几分钟的时间。

Amazon CloudFront KeyValueStore

CloudFront KeyValueStore 是一个安全、全局、低延迟的键值数据存储，允许从 CloudFront Functions 内部进行读取访问，从而在 CloudFront 边缘站点实现高级可自定义逻辑。

使用 CloudFront KeyValueStore，您可以对函数代码进行更新，并对与该函数关联的数据进行相互独立的更新。这种分离简化了函数代码，并且无需部署代码更改即可轻松更新数据。

> **Note**
>
> 要使用 CloudFront keyValueStore，您的 CloudFront 函数必须使用 JavaScript 运行时 2.0。

使用键值对的一般过程如下：

- 创建键值存储，并在其中填充一组键值对。您可以将键值存储添加到 Amazon S3 存储桶中，也可以手动输入。
- 将键值存储与您的 CloudFront 函数相关联。
- 在函数代码中，使用键的名称来检索与键关联的值或评估该键是否存在。有关在函数代码中使用键值对的更多信息以及有关助手方法的信息，请参阅the section called “键值存储的帮助程序方法”。

有关 CloudFront KeyValueStore 入门的更多信息，请参阅 Amazon CloudFront KeyValueStore AWS 博客文章。

您可以使用 CloudFront 控制台、CloudFront API 或受支持的 AWS SDK。要开始使用 CloudFront KeyValueStore，请参阅以下主题。

主题

使用案例

键值对的典型使用案例如下：

- URL 重写或重定向。键值对可以保存重写的 URL 或重定向 URL。
- A/B 测试和功能标志。您可以通过为网站的特定版本分配一定比例的流量来创建运行实验的函数。
- 访问授权。您可以实施访问控制，根据您定义的标准和存储在键值存储中的数据来允许或拒绝请求。

支持的值格式

键值对中的值可以采用以下任何一种格式进行存储：

- 字符串
- 字节编码的字符串
- JSON

安全性

CloudFront 函数及其所有键值存储数据均能得到安全处理，如下所示：

- CloudFront 会在您调用 CloudFront KeyValueStore API 操作时，在处于静态或在传输过程中（读取或写入键值存储时），对每个键值存储进行加密。
- 在函数运行时，CloudFront 会在 CloudFront 边缘站点对内存中的每个键值对进行解密。

使用键值存储

您必须创建一个键值存储来保存要在 CloudFront Functions 中使用的键值对。

创建键值存储并添加键值对后，即可在 CloudFront 函数代码中使用键值。JavaScript 运行时 2.0 包含一些用于在函数代码中处理键值的帮助程序方法。有关更多信息，请参阅 the section called “键值存储的帮助程序方法”。

主题

创建键值存储

您可以创建一个空的键值存储，稍后添加键值对。或者，您可以同时创建键值存储及其键值对。

> **Note**
>
> 如果您指定来自 Amazon S3 存储桶的数据来源，则必须对该存储桶具有 `s3:GetObject` 和 `s3:GetBucketLocation` 权限。如果您没有这些权限，CloudFront 将无法成功创建键值存储。

Console

创建键值存储（控制台）

1. 决定是否要在创建键值存储的同时添加键值对。CloudFront 控制台以及 CloudFront API 和 AWS SDK 都支持此导入功能。但仅在您最初创建键值存储时，才支持此功能。

   如果要使用文件，请立即创建它。
2. 登录到 AWS Management Console 并通过以下网址打开 CloudFront 控制台中的函数页面：https://console.aws.amazon.com/cloudfront/v4/home#/functions。
3. 选择 KeyValueStores 选项卡。选择创建 KeyValueStore。
4. 输入键值存储的名称和可选描述。
5. 填写 S3 URI：

   - 如果您已事先准备好一个键值对文件，请输入存储该文件的 Amazon S3 存储桶的路径。
   - 如果您打算手动输入键值对，请将此字段留空。
6. 选择创建。键值存储现已存在。

   此时将显示新键值存储的详细信息页面。页面上的信息包括键值存储的 ID 和 ARN。

   - ID 是一个随机字符串，在您的 AWS 账户中是唯一的。
   - ARN 的语法如下：

     ```
     AWS ##:key-value-store/#### ID
     ```
7. 查看键值对部分。如果您导入了文件，则此部分会显示一些键值对。否则为空。您可执行以下操作：

- 如果您没有从 Amazon S3 存储桶导入文件，并且想要立即添加键值对，则可以完成此部分。
- 如果您导入了文件，还可以手动添加更多值。
- 您可以将此部分留空，稍后再通过编辑键值存储来添加键值对。

立即添加键值对：

- 选择添加键值对按钮。
- 选择添加对并输入名称和值。
- 再次选择添加对按钮，以添加更多键值对。

完成后，选择保存更改，将所有键值对保存到键值存储中。在出现的确认对话框中，选择完成。

8. 如果您想立即将键值存储与函数关联，请完成关联的函数部分。您也可以稍后通过此键值存储详细信息页面或函数详细信息页面创建此关联。

要立即创建关联，请选择转至函数按钮。有关更多信息，请参阅???或???。

Programmatically

创建键值存储

1. 决定是否要在创建键值存储的同时添加键值对。（您也可以稍后添加键值对。）CloudFront 控制台以及 CloudFront API 和 SDK 都支持此导入功能。但仅在您最初创建键值存储时，才支持此功能。

   如果要使用文件，请立即创建它。

2. 使用 CloudFront API 或您首选的 AWS SDK 的创建操作。例如，对于 REST API，请使用 CloudFront.CreateKeyValueStore。该操作需要几个参数：
   - 名称。
   - 包含注释的 `configuration` 参数。
   - 一个 `import-source` 参数，允许您从存储在 Amazon S3 存储桶中的文件中导入键值对。请注意，只有在初次创建键值存储时，才能从文件导入。有关文件格式的信息，请参阅the section called “创建键值对的文件”。

操作响应包含以下信息：

- 请求中传递的值，包括您分配的名称。
- 诸如创建时间之类的数据。
- ETag（例如，ETVABCEXAMPLE2），即包含键值存储名称的 ARN（例如 `arn:aws:cloudfront::111122223333:key-value-store/MaxAge`）。

  您将使用 ETag、ARN 和名称的某种组合以编程方式使用键值存储。

键值存储状态

创建键值存储时，数据存储可以具有以下状态值。

值	描述
预置	键值存储已创建，CloudFront 正在处理您指定的数据来源。
就绪	键值存储已创建，CloudFront 成功处理了您指定的数据来源。
导入失败	CloudFront 无法处理您指定的数据来源。如果您的文件格式无效或超过大小限制，则会显示此状态。有关更多信息，请参阅 创建键值对的文件。

将键值存储与函数相关联

通过在函数中工作，可以将键值存储与函数相关联。必须进行此关联，才能在该函数中使用该存储中的键值对。以下规则适用：

- 一个函数可以有一个键值存储。
- 一个键值存储可以与多个函数相关联。

您可以通过以下方式使用关联。

- 您可以在函数和键值存储之间创建关联：
  - 在 CloudFront 控制台上，查看键值存储详细信息页面，然后选择转至函数按钮。将显示相应的页面：函数列表（如果当前没有关联的函数）或函数详细信息页面（如果当前存在关联）。有关更多信息，请参阅 the section called “将键值存储与函数相关联”。
  - 以编程方式，使用您首选的 CloudFront API 或 SDK 的函数更新操作。

创建关联后（或者如果您更改了关联），则应测试该函数，并且必须 重新发布该函数。

- 如果您在不更改键值对的情况下修改键值存储，则无需续订关联（这意味着您无需再次发布）。但您应该测试该函数。
- 如果您更改了键值存储中的键值对，则无需续订关联（这意味着您无需再次发布）。但您应该测试该函数，以验证它是否处理对键值对的更改。
- 您可以查看使用特定键值存储的所有函数。在 CloudFront 控制台上，查看键值存储详细信息页面。

修改键值存储

您可以使用键值对，并可以更改键值存储和函数之间的关联。

Console

修改键值存储

1. 登录到 AWS Management Console 并通过以下网址打开 CloudFront 控制台中的函数页面：https://console.aws.amazon.com/cloudfront/v4/home#/functions。
2. 选择 KeyValueStores 选项卡。选择要更改的键值存储。此时会显示详细信息页面。
   - 要使用键值对，请在键值对部分中选择编辑按钮。您可以添加更多键值对，可以删除任何键值对，也可以更改现有键值对的值。完成后，请选择保存更改。
   - 要使用此键值存储的关联，请选择转至函数按钮。将显示相应的页面：函数列表（如果当前没有关联的函数）或函数详细信息页面（如果当前存在关联）。有关更多信息，请参阅 the section called “将键值存储与函数相关联”。

Programmatically

您可以通过以下方式使用键值存储。

更改键值对

您可以添加更多键值对，可以删除一个或多个键值对，也可以更改现有键值对的值。有关更多信息，请参阅 the section called “以编程方式处理键值对”。

更改键值存储的函数关联

要使用此键值存储的关联，请参阅the section called “更新函数”。您将需要键值存储的 ARN。有关更多信息，请参阅 the section called “获取对键值存储的引用”。

删除键值存储

您可以使用 CloudFront 控制台或 API 来删除键值存储。

Console

删除键值存储

1. 登录到 AWS Management Console 并通过以下网址打开 CloudFront 控制台中的函数页面：https://console.aws.amazon.com/cloudfront/v4/home#/functions。
2. 验证键值存储是否与函数相关联。如果关联，则取消关联。有关这两个步骤的更多信息，请参阅???。
3. 选择 KeyValueStores 选项卡。选择要更改的键值存储，然后选择删除。

Programmatically

删除键值存储

1. 获取 ETag 和键值存储的名称。有关更多信息，请参阅 the section called “获取对键值存储的引用”。
2. 验证键值存储是否与函数相关联。如果关联，则取消关联。有关这两个步骤的更多信息，请参阅???。
3. 要删除键值存储，请使用您首选的 CloudFront API 或 SDK 的删除操作。例如，对于 REST API，请使用 CloudFront.DeleteKeyValueStore。

获取对键值存储的引用

为了以编程方式使用键值存储，您需要 ETag 和键值存储名称。要获取这些数据，可使用 CloudFront API 或您首选的 AWS SDK 并按照以下步骤操作：

1. 使用 CloudFront.ListKeyValueStores API 操作返回键值存储列表。找到要更改的键值存储的名称。
2. 使用 CloudFront.DescribeKeyValueStore API 操作并指定您在上一步中返回的键值存储的名称。

响应包括 UUID、键值存储的 ARN 和键值存储的 ETag。

- UUID 为 128 位。例如，a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
- ARN 包括 AWS 账户编号、常量 key-value-store 和 UUID。例如：

  arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
- ETag 类似于 ETVABCEXAMPLE2

有关 DescribeKeyValueStore 操作的更多信息，请参阅the section called “关于 CloudFront KeyValueStore”。

创建键值对的文件

创建 UTF-8 编码文件时，请使用以下 JSON 格式：

```
{
  "data":[
    {
      "key":"key1",
      "value":"value"
    },
    {
      "key":"key2",
      "value":"value"
    }
  ]
}
```

您的文件不能包含重复键。如果您在 Amazon S3 存储桶中指定了无效文件，则可以更新该文件以删除所有重复项，然后再次尝试创建键值存储。

有关更多信息，请参阅 创建键值存储。

> **Note**
>
> 您的数据来源文件及其键值对具有以下限制：
>
> - 文件大小 – 5 MB
> - 键大小 – 512 个字符
> - 值大小 – 1024 个字符

处理键值数据

您可以通过以下方式使用现有键值存储中的键值对：

- 使用 Amazon CloudFront 控制台。
- 使用 CloudFront KeyValueStore API 或您首选的 AWS SDK。

本节介绍如何将键值对添加到现有的键值存储中。要在最初创建键值存储时包含键值对，请参阅the section called “创建键值存储”。

主题

- 使用 CloudFront 控制台处理键值对
- 以编程方式处理键值对

使用 CloudFront 控制台处理键值对

您可以使用 CloudFront 控制台处理键值对。

使用键值对

1. 登录到 AWS Management Console 并通过以下网址打开 CloudFront 控制台中的函数页面：https://console.aws.amazon.com/cloudfront/v4/home#/functions。
2. 选择 KeyValueStores 选项卡。选择要更改的键值存储。此时会显示详细信息页面。
3. 在键值对部分，选择编辑。
4. 您可以添加键值对、删除键值对或更改现有键值对的值。
5. 完成后，请选择保存更改。

以编程方式处理键值对

> **Note**
>
> CloudFront KeyValueStore API 的命名空间与 CloudFront API 的命名空间不同。

主题

- 获取对键值存储的引用

- 更改键值存储中的键值对
- 关于 CloudFront KeyValueStore
- CloudFront KeyValueStore 代码示例

获取对键值存储的引用

当您使用 CloudFront KeyValueStore 输入写入操作时，需要传入键值存储的 ARN 和 ETag。要获取此数据，请执行以下操作：

1. 使用您首选的 CloudFront API 或 SDK 的列表操作。例如，对于 REST API，请使用 CloudFront.ListKeyValueStores。响应包含键值存储的列表。找到要更改的键值存储的名称。
2. 使用您首选的 CloudFront KeyValueStore API 或 SDK 的描述操作。例如，对于 REST API，请使用 CloudFrontKeyValueStore.DescribeKeyValueStore。传入您在上一步中获得的名称。

> **Note**
>
> 使用 CloudFront KeyValueStore API 中的操作，而不是来自 CloudFront API 的操作。有关更多信息，请参阅 the section called “关于 CloudFront KeyValueStore”。

   响应包括键值存储的 ARN 和 ETag。

   - ARN 包括 AWS 账户编号、常量 `key-value-store` 和 UUID。例如：

     ```
     arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
     ```

   - ETag 类似于 `ETVABCEXAMPLE2`

更改键值存储中的键值对

您可以使用首选的 CloudFront KeyValueStore API 或 SDK 的以下操作来处理键值对。所有这些操作都适用于一个指定的键值存储：

- `CloudFrontKeyValueStore.DeleteKey`：删除一个键。请参阅 DeleteKey。
- `CloudFrontKeyValueStore.GetKey`：获取一个键。请参阅 GetKey。
- `CloudFrontKeyValueStore.ListKeys`：列出键。请参阅 ListKeys。

- `CloudFrontKeyValueStore.PutKey`：您可以执行两项操作：
  - 在一个键值存储中创建新的键值对：在这种情况下，将传递新的键名称和值。
  - 在一个现有键值对中设置不同的值：在这种情况下，将传递一个现有键名称和一个新的键值。

  请参阅 PutKey。
- `CloudFrontKeyValueStore.UpdateKeys`：您可以通过一项“要么全有，要么全无”操作执行以下一项或多项操作：
  - 删除一个或多个键值对。
  - 创建一个或多个新的键值对。
  - 在一个或多个现有键值对中设置不同的值。

  请参阅 UpdateKeys。

关于 CloudFront KeyValueStore

要在现有 键值存储中以编程方式处理键值对，您可以使用 CloudFront KeyValueStore 服务。

要在最初创建键值存储时在键值存储中包含一些键值对，可以使用 CloudFront 服务。

描述操作

CloudFront API 和 CloudFront KeyValueStore API 都有一个描述操作，可以返回有关键值存储的数据：

- CloudFront API 提供诸如状态和存储本身上次修改日期之类的数据。
- CloudFront KeyValueStore API 提供有关存储资源的内容 的数据，即存储中的键值对以及内容的大小。

这两个 API 中的描述操作返回的标识键值存储的数据略有不同：

- CloudFront API 中的描述操作会返回键值存储的 ETag、UUID 和 ARN。

- CloudFront KeyValueStore API 中的描述操作返回键值存储的 ETag 和 ARN。

> **Note**
>
> 每个描述操作都会返回一个不同 的 ETag。ETag 不可互换。

当您在其中一个 API 中执行操作时，必须从相应的 API 中传入 ETag。例如，在 CloudFront KeyValueStore 的删除操作中，传入您从 CloudFront KeyValueStore 中的描述操作获得的 ETag。

CloudFront KeyValueStore 代码示例

Example ：调用 **`DescribeKeyValueStore`** API 操作

以下代码示例演示了如何为键值存储调用 `DescribeKeyValueStore` API 操作。

```
const {
  CloudFrontKeyValueStoreClient,
  DescribeKeyValueStoreCommand,
} = require("@aws-sdk/client-cloudfront-keyvaluestore");

require("@aws-sdk/signature-v4-crt");

(async () => {
  try {
    const client = new CloudFrontKeyValueStoreClient({
      region: "us-east-1"
    });
    const input = {
      KvsARN: "arn:aws:cloudfront::123456789012:key-value-store/a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111",
    };
    const command = new DescribeKeyValueStoreCommand(input);

    const response = await client.send(command);
  } catch (e) {
    console.log(e);
  }
})();
```

使用 Lambda@Edge 在边缘进行自定义

Lambda@Edge 是 AWS Lambda 的扩展。Lambda@Edge 是一项计算服务，可用于执行函数以自定义 Amazon CloudFront 提供的内容。您可以在某个 AWS 区域，比如美国东部（弗吉尼亚州北部）的 Lambda 控制台中编写 Node.js 或 Python 函数。

然后，您可以在 Lambda 或 CloudFront 控制台中添加触发器，使函数能够在与查看器较为接近的 AWS 位置运行，而无需预置或管理服务器。或者，您也可以使用 Lambda 和 CloudFront API 操作以编程方式设置函数和触发器。

Lambda@Edge 会自动扩展，从每天几个请求到每秒数千个请求。在与查看器较为接近的 AWS 位置（而不是源服务器）上处理请求，可显著减少延迟并改善用户体验。

主题

了解 Lambda@Edge 如何处理请求和响应

在将 CloudFront 分配与 Lambda@Edge 函数相关联时，CloudFront 在 CloudFront 边缘站点中截获请求和响应。当发生以下 CloudFront 事件时，您可以执行 Lambda 函数：

- 在 CloudFront 收到查看器的请求时 (查看器请求)
- 在 CloudFront 将请求转发到源之前（源请求）
- 在 CloudFront 收到来自源的响应时（源响应）
- 在 CloudFront 将响应返回到查看器之前（查看器响应）

如果您使用的是 AWS WAF，则会在应用任何 AWS WAF 规则后执行 Lambda@Edge 查看器请求。

有关更多信息，请参阅使用请求和响应和 Lambda@Edge 事件结构。

使用 Lambda@Edge 的方法

在 Amazon CloudFront 分配中，Lambda@Edge 处理有很多用途。例如：

- Lambda 函数可检查 Cookie 并重写 URL，以便用户可看到不同版本的站点以进行 A/B 测试。
- CloudFront 可通过检查 `User-Agent` 标头来基于查看器使用的设备将不同的对象返回到查看器，该标头包含有关这些设备的信息。例如，CloudFront 可基于不同图像所在的设备的屏幕尺寸返回这些图像。同样，函数可考虑 `Referer` 标头的值，并使 CloudFront 将图像返回到具有最低的可用分辨率的自动程序。
- 或者，您也可以检查 Cookie 中是否有其他条件。例如，在出售服装的零售网站上，如果您使用 Cookie 来指示用户选择了哪种颜色的夹克，Lambda 函数可更改相应请求，以便 CloudFront 返回选定颜色的夹克的图像。
- 在发生 CloudFront 查看器请求或源请求事件时，Lambda 函数可生成 HTTP 响应。
- 函数可检查标头或授权令牌，并在 CloudFront 将请求转发到您的源之前插入一个标头，以控制对您的内容的访问。
- Lambda 函数还可以向外部资源发出网络调用，以确认用户凭证，或获取更多内容来自定义响应。

有关更多用途（包括代码示例），请参阅 Lambda@Edge 函数示例。

有关向您展示如何在控制台中设置 Lambda@Edge 的过程，请参阅教程：创建基本 Lambda@Edge 函数。

Lambda@Edge 函数入门

借助 Lambda@Edge，您可以使用 CloudFront 触发器调用 Lambda 函数。在将 CloudFront 分配与 Lambda 函数相关联时，CloudFront 在 CloudFront 边缘站点中截获请求和响应并运行函数。Lambda 函数可以提高安全性，或者自定义靠近查看器的信息以提高性能。

下表概述了如何创建 Lambda 函数并将其用于 CloudFront。如需分步教程，请参阅教程：创建基本 Lambda@Edge 函数。

1. 在 AWS Lambda 控制台中，在美国东部（弗吉尼亚州北部）区域创建 Lambda 函数。（或者，您可以使用某个 AWS SDK 以编程方式创建函数）。
2. 保存和发布带编号的函数版本。

   如果要对函数进行更改，则必须在美国东部（弗吉尼亚州北部）区域中编辑函数的 $LATEST 版本。然后，在将其设置为与 CloudFront 结合使用之前，发布一个带编号的新版本。

3. 将函数与 CloudFront 分配和缓存行为进行关联。指定触发函数执行的一个或多个 CloudFront 事件（称作触发器）。例如，您可以创建一个在 CloudFront 收到查看器的请求时促使函数执行的触发器。
4. 创建触发器时，Lambda 会在全球各个 AWS 位置创建该函数的副本。

> **Tip**
>
> 详细了解如何使用 Lambda@Edge 创建您自己的自定义解决方案。了解有关创建和更新函数、事件结构和添加 CloudFront 触发器的更多信息。另外，您可以在 Lambda@Edge 函数示例 中找到更多创意并获得代码示例。

主题

教程：创建基本 Lambda@Edge 函数

本教程将演示如何通过创建和配置一个在 CloudFront 中运行的简单 Node.js 函数，来开始使用 Lambda@Edge。在该示例中，我们在 CloudFront 检索文件时将 HTTP 安全标头添加到响应中。（这可以提高网站的安全性和隐私性。）

学习本教程不需要您自己的网站。但在选择创建自己的 Lambda@Edge 解决方案时，您需要完成类似步骤并从相同的选项中进行选择。

主题

第 1 步：注册 AWS 账户

如果您尚未完成此操作，请注册一个 AWS 账户。有关更多信息，请参阅 注册 AWS 账户。

步骤 2：创建 CloudFront 分配

创建 Lambda@Edge 函数示例之前，您必须有一个可使用的、包含提供内容的源的 CloudFront 环境。

在本示例中，您将创建一个使用 Amazon S3 存储桶作为分配源的 CloudFront 分配。如果您已有要使用的环境，则可跳过本步骤。

创建具有 Amazon S3 源的 CloudFront 分配

1. 创建包含一个或两个文件的 Amazon S3 存储桶，此处的文件可以是图像文件等等，并将作为示例内容。要获得帮助，请按照将您的内容上传到 Amazon S3 中提供的步骤执行操作。确保设置相应权限，以授予对存储桶中对象的公共读取访问权限。
2. 按照创建 CloudFront Web 分配中提供的步骤操作，创建 CloudFront 分配并添加您的 S3 存储桶作为源。如果您已有一个分配，可以添加该存储桶作为该分配的源。

> **Tip**
>
> 请记下您的分配 ID。在本教程的后面，为函数添加 CloudFront 触发器时必须从下拉列表中为您的分配选择该 ID，例如 `E653W22221KDDL`。

第 3 步：创建函数

在此步骤中，您将从 Lambda 控制台中的蓝图模板创建一个 Lambda 函数。该函数会添加代码以更新 CloudFront 分配中的安全标头。

创建 Lambda 函数

1. 通过以下网址登录 AWS Management Console并打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。

> **Important**
>
> 确保您位于 US-East-1（弗吉尼亚州北部）AWS 区域（us-east-1）。您必须位于该区域，才能创建 Lambda@Edge 函数。

2. 选择创建函数。
3. 在创建函数页面上，选择使用蓝图，然后通过在搜索字段中输入 **`cloudfront`** 来筛选 CloudFront 蓝图。

> **Note**
>
> CloudFront 蓝图仅在 US-East-1（弗吉尼亚州北部）区域（us-east-1）可用。

4. 选择修改 HTTP 响应标头蓝图以用作函数的模板。
5. 输入有关函数的以下信息：

   函数名称

   输入您的函数的名称。

   执行角色

   选择如何设置函数的权限。要使用推荐的基本 Lambda@Edge 权限策略模板，请选择从 AWS 策略模板创建新角色。

   角色名称

   输入策略模板所创建的角色的名称。

   策略模板

   Lambda 将自动添加策略模板基本 Lambda@Edge 权限，因为您选择 CloudFront 蓝图作为您函数的基础。此策略模板添加执行角色权限，以允许 CloudFront 在全球各地的 CloudFront 位置运行您的 Lambda 函数。有关更多信息，请参阅 设置 Lambda@Edge 的 IAM 权限和角色。
6. 选择创建函数。
7. 在出现的部署到 Lambda@Edge 窗格中，选择取消。（对于本教程，在将函数部署到 Lambda@Edge 之前，您必须修改函数代码。）
8. 向下滚动到页面的代码源部分。
9. 将模板代码替换为一个函数，该函数修改您的源返回的安全标头。例如，可以使用如下代码：

```
'use strict';
exports.handler = (event, context, callback) => {

    //Get contents of response
    const response = event.Records[0].cf.request;
    const headers = response.headers;
```

```
    //Set new headers
    headers['strict-transport-security'] = [{key: 'Strict-Transport-Security',
 value: 'max-age= 63072000; includeSubdomains; preload'}];
    headers['content-security-policy'] = [{key: 'Content-Security-Policy', value:
 "default-src 'none'; img-src 'self'; script-src 'self'; style-src 'self'; object-
src 'none'"}];
    headers['x-content-type-options'] = [{key: 'X-Content-Type-Options', value:
 'nosniff'}];
    headers['x-frame-options'] = [{key: 'X-Frame-Options', value: 'DENY'}];
    headers['x-xss-protection'] = [{key: 'X-XSS-Protection', value: '1;
 mode=block'}];
    headers['referrer-policy'] = [{key: 'Referrer-Policy', value: 'same-origin'}];

    //Return modified response
    callback(null, response);
};
```

10. 选择文件、保存，保存已更新的代码。

继续执行下一部分，添加 CloudFront 触发器以运行函数。

第 4 步：添加 CloudFront 触发器来运行函数

现已具有用于更新安全标头的 Lambda 函数，请配置 CloudFront 触发器以运行您的函数，从而在 CloudFront 从分配来源收到的任何响应中添加标头。

为您的函数配置 CloudFront 触发器

1. 在 Lambda 控制台中，在函数的函数概述页面上，选择添加触发器。
2. 对于触发器配置，请选择 CloudFront。
3. 选择部署到 Lambda@Edge。
4. 在部署到 Lambda@Edge窗格上的配置 CloudFront 触发器下，输入以下信息：

   分配

   要与函数关联的 CloudFront 分配 ID。从下拉列表中选择分配 ID。

   缓存行为

   要用于触发器的缓存行为。在本示例中，将该值设置为 *，这表示您的分配的默认缓存行为。有关更多信息，请参阅分配设置参考主题中的 缓存行为设置。

CloudFront 事件

指定何时运行您的函数的触发器。我们希望在 CloudFront 返回来自源的响应时运行安全标头函数。因此，在下拉列表中选择源响应。有关更多信息，请参阅 为 Lambda@Edge 函数添加触发器。

5. 选中确认部署到 Lambda@Edge复选框。
6. 选择部署以添加触发器并将该函数复制到全球各地的 AWS 位置。
7. 请等待复制函数完成。这通常需要几分钟时间。

   您可以转到 CloudFront 控制台并查看您的分配，确认复制是否完成。等待分配状态从正在部署更改为部署日期和时间，这意味着已复制函数。要验证函数是否正常工作，请执行下一节中的步骤。

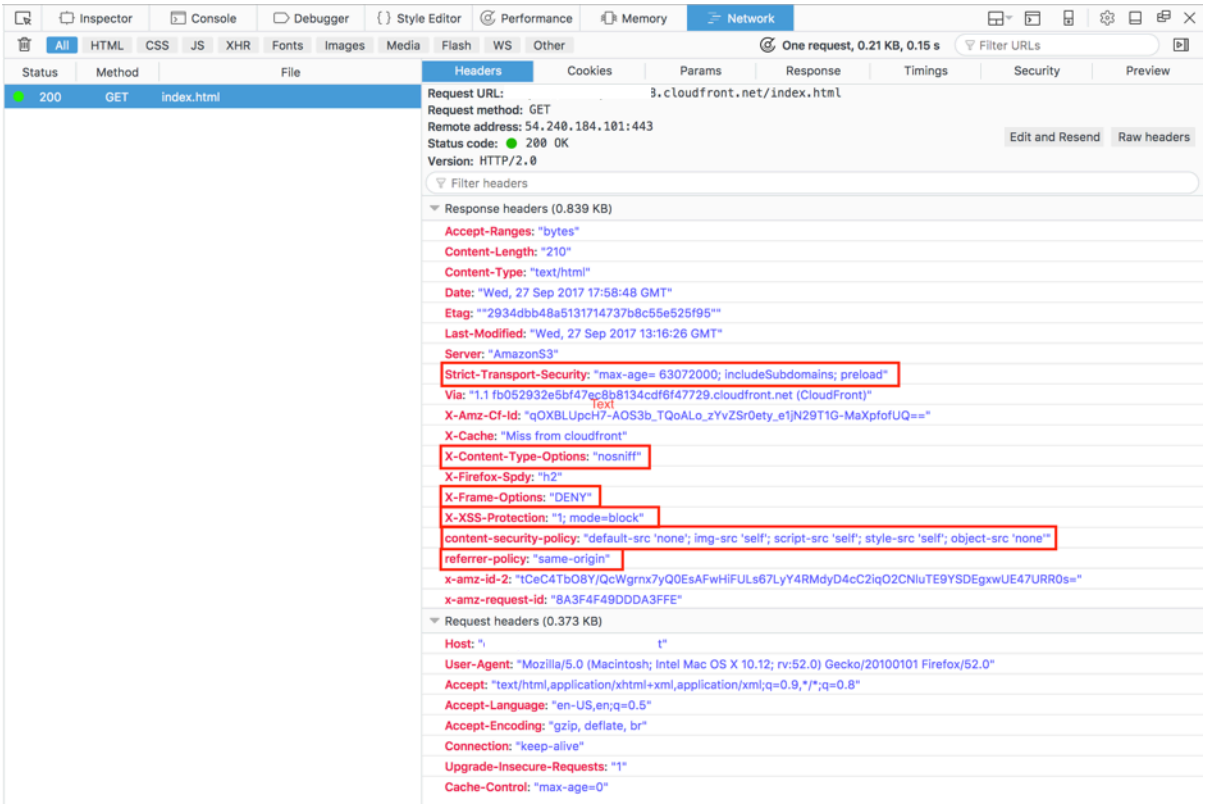
第 5 步：验证函数是否正常工作

现在，您已创建 Lambda 函数并已配置触发器以便为 CloudFront 分配运行它，请检查以确保该函数按照预期运行。在本示例中，我们检查 CloudFront 返回的 HTTP 标头，确保已添加了安全标头。

验证您的 Lambda@Edge 函数是否添加安全标头

1. 在浏览器中，键入 S3 存储桶中的文件的 URL。例如，您可以使用类似以下所示的 URL：`https://d111111abcdef8.cloudfront.net/image.jpg`。

   有关要在文件 URL 中使用的 CloudFront 域名的更多信息，请参阅在 CloudFront 中自定义文件的 URL 格式。
2. 打开您的浏览器的 Web Developer（Web 开发人员）工具栏。例如，在 Chrome 浏览器窗口中，打开上下文（右键单击）菜单，然后选择检查。
3. 选择网络选项卡。
4. 重新加载页面以查看您的图像，然后在左侧窗格选择 HTTP 请求。您会看到在一个单独窗格中显示 HTTP 标头。
5. 查看 HTTP 标头的列表，验证所需安全标头包含在列表中。例如，您可能会看到类似于以下屏幕截图中所示的标头。



如果安全标头已包含在标头列表中，很棒！您已成功创建第一个 Lambda@Edge 函数。如果 CloudFront 返回错误或有其他问题，请继续执行下一步骤来解决相应问题。

第 6 步：问题排查

如果 CloudFront 返回错误或没有按预期添加安全标头，您可以查看 CloudWatch Logs 来研究函数的执行。请务必使用与执行函数时的位置最接近的 AWS 位置中存储的日志。

例如，如果您查看来自伦敦的文件，请尝试在 CloudWatch 控制台中将“区域”更改为“欧洲地区（伦敦）”。

检查您的 Lambda@Edge 函数的 CloudWatch 日志

1. 登录到 AWS Management Console 并通过以下网址打开 CloudWatch 控制台：https://console.aws.amazon.com/cloudwatch/。
2. 将区域更改为您在浏览器中查看文件时所显示的位置。这是执行函数的区域。
3. 在左侧窗格中选择日志以查看您的分配的日志。

有关更多信息，请参阅 使用 Amazon CloudWatch 监控 CloudFront 指标。

第 7 步：清除示例资源

如果您仅为本教程创建了 Amazon S3 存储桶和 CloudFront 分配，请删除您分配的 AWS 资源，以免继续产生费用。删除 AWS 资源后，您添加的任何内容将不再可用。

任务

- 删除 S3 存储桶
- 删除 Lambda 函数
- 删除 CloudFront 分配

删除 S3 存储桶

在删除您的 Amazon S3 存储桶之前，请确保已禁用该存储桶的日志记录。否则，在您删除存储桶时，AWS 会继续将日志写入其中。

禁用存储桶的日志记录

1. 通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择存储桶，然后选择 Properties。
3. 从属性中选择日志记录。
4. 取消选中启用复选框。
5. 选择保存。

现在您可以删除存储桶。有关更多信息，请参阅《Amazon Simple Storage Service 控制台用户指南》中的删除存储桶。

删除 Lambda 函数

有关如何取消 Lambda 函数关联以及选择删除函数本身的说明，请参阅删除 Lambda@Edge 函数和副本。

删除 CloudFront 分配

删除 CloudFront 分配之前，须先将其禁用。已禁用的分发不再起作用，并且不会产生费用。您可以随时启用已禁用的分发。已禁用的分配在删除后将不再可用。

禁用并删除 CloudFront 分配

1. 通过 打开 CloudFront 控制台https://console.aws.amazon.com/cloudfront/v4/home

2. 选择要禁用的分配，然后选择禁用。
3. 当系统提示确认时，选择是，禁用。
4. 选择禁用的分配，然后选择删除。
5. 当系统提示进行确认时，选择 Yes, Delete。

可了解更多信息的资源

现在，您对 Lambda@Edge 函数的工作方式已经有了基本概念，如需了解更多信息，请阅读以下内容：

- Lambda@Edge 函数示例
- Lambda@Edge 设计最佳实践
- 使用 Lambda@Edge 减少延迟并将计算转移到边缘站点

设置 Lambda@Edge 的 IAM 权限和角色

要配置 Lambda@Edge，您必须针对 Lambda 设置以下 IAM 权限和角色：

- IAM 权限 – 这些权限允许您创建自己的 AWS Lambda 函数并将其与您的 CloudFront 分配相关联。
- Lambda 函数执行角色（IAM 角色）– Lambda 服务主体代入此角色来执行您的函数。
- 服务相关 Lambda@Edge 角色 – 服务相关角色允许特定 AWS 服务将 Lambda 函数复制到 AWS 区域，并允许 CloudWatch 使用 CloudFront 日志文件。

将 Lambda@Edge 函数与 CloudFront 分配关联所需的 IAM 权限

除了配置 Lambda 所需的 IAM 权限之外，您还需要以下权限才能将 Lambda 函数与 CloudFront 分配相关联：

- `lambda:GetFunction` – 授予相关权限，已获取 Lambda 函数的配置信息，以及一个用于下载包含该函数的 `.zip` 文件的预签名 URL。
- `lambda:EnableReplication*` – 向资源策略授予相关权限，以便 Lambda 复制服务可以获取函数代码和配置。
- `lambda:DisableReplication*` – 向资源策略授予相关权限，以便 Lambda 复制服务可以删除函数。

> **Important**
>
> 您必须在 `lambda:EnableReplication*` 和 `lambda:DisableReplication*` 操作的末尾添加星号（`*`）。

- 对于资源，请指定当 CloudFront 事件发生时要执行的函数版本的 ARN，如以下示例所示：

  `arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2`

- `iam:CreateServiceLinkedRole` – 授予相关权限，以允许创建 Lambda@Edge 用于在 CloudFront 中复制 Lambda 函数所需的服务相关角色。在首次配置 Lambda@Edge 之后，将自动创建服务相关角色。您不需要将此权限添加至使用 Lambda@Edge 的其他分配中。

- `cloudfront:UpdateDistribution` 或 `cloudfront:CreateDistribution` - 授予更新或创建分配的权限。

有关更多信息，请参阅以下主题：

- 适用于 Amazon CloudFront 的 Identity and Access Management
- 《AWS Lambda 开发人员指南》中的 Lambda 资源访问权限

服务主体的函数执行角色

您必须创建一个 IAM 角色，以便 `lambda.amazonaws.com` 和 `edgelambda.amazonaws.com` 服务主体在执行您的函数时可以代入该角色。

> **Tip**
>
> 当您在 Lambda 控制台中创建函数时，可以选择使用 AWS 策略模板创建新的执行角色。此步骤会自动 添加执行函数所需的 Lambda@Edge 权限。请参阅教程中的步骤 5：创建简单的 Lambda@Edge 函数。

有关手动创建 IAM 角色的更多信息，请参阅《IAM 用户指南》中的创建角色并附加策略（控制台）。

Example 示例：角色信任策略

您可以在 IAM 控制台的信任关系选项卡下添加此角色。请勿在权限选项卡下添加此策略。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "Service": [
               "lambda.amazonaws.com",
               "edgelambda.amazonaws.com"
            ]
         },
         "Action": "sts:AssumeRole"
      }
   ]
}
```

有关需要向执行角色授予的权限的更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 资源访问权限。

> **注意**
>
> - 默认情况下，每当 CloudFront 事件触发 Lambda 函数时，数据都会写入到 CloudWatch Logs。如果要使用这些日志，执行角色需要权限来将数据写入 CloudWatch Logs。您可以使用预定义的 AWSLambdaBasicExecutionRole 向执行角色授予权限。
>
> 有关 CloudWatch Logs 的更多信息，请参阅the section called “边缘函数日志”。
> - 如果您的 Lambda 函数代码访问其他 AWS 资源，比如从 S3 存储桶读取对象，则执行角色需要权限来执行此操作。

Lambda@Edge 的服务相关角色

Lambda@Edge 使用 IAM 服务相关角色。服务相关角色是一种与服务直接关联的独特类型的 IAM 角色。服务相关角色是由服务预定义的，具有服务代表您调用其他 AWS 服务所需的所有权限。

Lambda@Edge 使用以下 IAM 服务相关角色：

- AWSServiceRoleForLambdaReplicator – Lambda@Edge 使用该角色来允许 Lambda@Edge 将函数复制到 AWS 区域。

当您首次在 CloudFront 中添加 Lambda@Edge 触发器时，会自动创建一个名为 AWSServiceRoleForLambdaReplicator 的角色，以允许 Lambda@Edge 将函数复制到 AWS 区域。使用 Lambda@Edge 函数也需要该角色。例如，AWSServiceRoleForLambdaReplicator 角色的 ARN 如下所示：

```
arn:aws:iam::123456789012:role/aws-service-role/
replicator.lambda.amazonaws.com/AWSServiceRoleForLambdaReplicator
```

- AWSServiceRoleForCloudFrontLogger – CloudFront 使用此角色将日志文件推送到 CloudWatch。您可以使用日志文件来调试 Lambda@Edge 验证错误。

  在添加 Lambda@Edge 函数关联以允许 CloudFront 将 Lambda@Edge 错误日志文件推送到 CloudWatch 时，将自动创建 AWSServiceRoleForCloudFrontLogger 角色。AWSServiceRoleForCloudFrontLogger 角色的 ARN 如下所示：

  ```
  arn:aws:iam::account_number:role/aws-service-role/
  logger.cloudfront.amazonaws.com/AWSServiceRoleForCloudFrontLogger
  ```

通过使用服务相关角色，您可以更轻松地设置和使用 Lambda@Edge，因为您不必手动添加所需的权限。Lambda@Edge 定义其服务相关角色的权限，并且仅 Lambda@Edge 可以担任该角色。定义的权限包括信任策略和权限策略。不能将该权限策略附加到任何其他 IAM 实体。

您必须先删除任何关联的 CloudFront 或 Lambda@Edge 资源，然后才能删除服务相关角色。这有助于保护您的 Lambda@Edge 资源，使您不会删除访问活动资源时仍需要的服务相关角色。

有关服务相关角色的更多信息，请参阅CloudFront 的服务相关角色。

Lambda@Edge 的服务相关角色权限

Lambda@Edge 使用两个名为 AWSServiceRoleForLambdaReplicator 和 AWSServiceRoleForCloudFrontLogger 的服务相关角色。以下部分介绍了其中的每个角色的权限。

目录

Lambda Replicator 的服务相关角色权限

此服务相关角色允许 Lambda 将 Lambda@Edge 函数复制到AWS 区域。

AWSServiceRoleForLambdaReplicator 服务相关角色信任 `replicator.lambda.amazonaws.com` 服务来代入角色。

角色权限策略允许 Lambda@Edge 对指定的资源完成以下操作：

- `lambda:CreateFunction`，发布时间：`arn:aws:lambda:*:*:function:*`
- `lambda:DeleteFunction`，发布时间：`arn:aws:lambda:*:*:function:*`
- `lambda:DisableReplication`，发布时间：`arn:aws:lambda:*:*:function:*`
- `iam:PassRole`，发布时间：`all AWS resources`
- `cloudfront:ListDistributionsByLambdaFunction`，发布时间：`all AWS resources`

CloudFront Logger 的服务相关角色权限

该服务相关角色允许 CloudFront 将日志文件推送到 CloudWatch 账户，以便您可以调试 Lambda@Edge 验证错误。

AWSServiceRoleForCloudFrontLogger 服务相关角色信任 `logger.cloudfront.amazonaws.com` 服务来代入角色。

该角色权限策略允许 Lambda@Edge 对指定的 `arn:aws:logs:*:*:log-group:/aws/cloudfront/*` 资源执行以下操作：

- `logs:CreateLogGroup`
- `logs:CreateLogStream`
- `logs:PutLogEvents`

您必须配置权限以允许 IAM 实体（如用户、组或角色）删除 Lambda@Edge 服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

为 Lambda@Edge 创建服务相关角色

通常您不需要为 Lambda@Edge 手动创建服务相关角色。在以下情况下，该服务自动为您创建角色：

- 在首次创建触发器时，该服务会创建一个 AWSServiceRoleForLambdaReplicator 角色（如果该角色尚不存在）。该角色允许 Lambda 将 Lambda@Edge 函数复制到 AWS 区域。

  如果您删除服务相关角色，则在分配中为 Lambda@Edge 添加新触发器时，将再次创建该角色。

- 在更新或创建具有 Lambda@Edge 关联的 CloudFront 分配时，该服务会创建 AWSServiceRoleForCloudFrontLogger 角色（如果该角色尚不存在）。该角色允许 CloudFront 将日志文件推送到 CloudWatch。

  如果删除服务相关角色，在更新或创建具有 Lambda@Edge 关联的 CloudFront 分配时，将再次创建该角色。

要手动创建这些服务相关角色，可以运行以下 AWS Command Line Interface（AWS CLI）命令：

创建 AWSServiceRoleForLambdaReplicator 角色

- 运行以下命令。

```
aws iam create-service-linked-role --aws-service-name
 replicator.lambda.amazonaws.com
```

创建 AWSServiceRoleForCloudFrontLogger 角色

- 运行以下命令。

```
aws iam create-service-linked-role --aws-service-name
 logger.cloudfront.amazonaws.com
```

编辑 Lambda@Edge 服务相关角色

Lambda@Edge 不允许您编辑 AWSServiceRoleForLambdaReplicator 或 AWSServiceRoleForCloudFrontLogger 服务相关角色。在该服务创建服务相关角色后，您无法更改该角色的名称，因为不同的实体可能会引用该角色。但是，您可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》 中的编辑服务相关角色。

CloudFront 服务相关角色支持的AWS 区域

CloudFront 支持在以下AWS 区域对 Lambda@Edge 使用服务相关角色：

- 美国东部（弗吉尼亚州北部）– us-east-1
- 美国东部（俄亥俄州）– us-east-2
- 美国西部（加利福尼亚北部）– us-west-1
- 美国西部（俄勒冈州）– us-west-2

- 亚太地区（孟买）– (ap-south-1)
- 亚太地区（首尔）– (ap-northeast-2)
- 亚太地区（新加坡）– (ap-southeast-1)
- 亚太地区（悉尼）– ap-southeast-2
- 亚太地区（东京）– (ap-northeast-1)
- 欧洲地区（法兰克福）– eu-central-1
- 欧洲地区（爱尔兰）– eu-west-1
- 欧洲地区（伦敦）– eu-west-2
- 南美洲（圣保罗）– (sa-east-1)

编写和创建 Lambda@Edge 函数

要使用 Lambda@Edge，您需要为 AWS Lambda 函数编写 代码。接下来，您需要设置 Lambda 以基于特定的 CloudFront 事件（称为触发器）运行该函数。

您可以通过 AWS Management Console 使用 Lambda 函数和 CloudFront 触发器，或者也可以通过 API 以编程方式使用 Lambda@Edge。

主题

编写 Lambda@Edge 函数

为了帮助您编写 Lambda@Edge 函数，请参阅以下资源：

- Lambda@Edge 事件结构 – 了解可用于 Lambda@Edge 的事件结构。
- Lambda@Edge 函数示例 – 函数示例（例如 A/B 测试和生成 HTTP 重定向）。

将 Node.js 或 Python 用于 Lambda@Edge 的编程模型与在 AWS 区域内使用 Lambda 的编程模型相同。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的使用 Node.js 构建 Lambda 函数或使用 Python 构建 Lambda 函数。

在 Lambda@Edge 函数中，包含 `callback` 参数并返回适用于请求或响应事件的对象：

- 请求事件 – 在响应中包含 `cf.request` 对象。

  如果要生成响应，请在响应中包含 `cf.response` 对象。有关更多信息，请参阅 在请求触发器中生成 HTTP 响应。
- 响应事件 – 在响应中包含 `cf.response` 对象。

创建 Lambda@Edge 函数

要将 AWS Lambda 设置为运行基于 CloudFront 事件的 Lambda 函数，请按照以下步骤操作。

创建 Lambda@Edge 函数（控制台）

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 如果您已有一个或多个 Lambda 函数，请选择创建函数。

   如果您没有任何函数，请选择立即开始使用。
3. 在页面顶部的“区域”列表中，选择美国东部（弗吉尼亚州北部）。
4. 使用您自己的代码创建函数，或者以 CloudFront 蓝图为基础创建函数。

   - 要使用您自己的代码创建函数，请选择从头开始创作。
   - 要显示 CloudFront 蓝图列表，请在筛选条件字段中键入 cloudfront，然后选择 Enter。

     如果您找到了自己要使用的蓝图，请选择该蓝图的名称。
5. 在基本信息部分，指定以下值：

   a. 名称 – 输入函数的名称。

   b. 角色 – 要快速入门，请选择从模板创建新角色。您也可以选择选择现有角色或创建自定义角色，然后按照提示填写本部分的信息。

   c. 角色名称 – 输入角色的名称。

   d. 策略模板 – 选择基本 Edge Lambda 权限。
6. 如果您在步骤 4 中选择了从头开始创作，请跳至步骤 7。

   如果您在步骤 4 中选择了蓝图，则可通过 cloudfront 部分创建一个触发器，它可将此函数与 CloudFront 分配和 CloudFront 事件中的缓存相关联。建议您现在选择删除，因此在创建函数时没有函数触发器。您可以在稍后添加触发器。

> **Tip**
>
> 建议您先测试和调试该函数，然后再添加触发器。如果选择立即添加触发器，则在您创建该函数，该函数完成向全球 AWS 位置的复制，并且相应的分配部署完成后，该函数将立即开始运行。

7. 选择创建函数。

   Lambda 将创建两个版本的函数：$LATEST 和 Version 1。您只能编辑 $LATEST 版本，但控制台最初会显示 Version 1。
8. 要编辑函数，请选择页面顶部附近、函数 ARN 下方的 Version 1。然后，在 Versions 选项卡上，选择 $LATEST。（如果您离开再返回到该函数，则按钮标签将是 Qualifiers。）
9. 在 Configuration 选项卡上，选择适用的 Code entry type。然后，按照提示编辑或上传您的代码。
10. 对于运行时，请根据函数的代码选择值。
11. 在标签部分中，添加任何适用的标签。
12. 选择操作，然后选择发布新版本。
13. 键入新版本函数的说明。
14. 选择 Publish。
15. 测试并调试函数。有关在 Lambda 控制台中进行测试的更多信息，请参阅《AWS Lambda 开发人员指南》中的使用控制台创建 Lambda 函数中的调用 Lambda 函数和验证结果、日志和指标部分。
16. 当您准备好为 CloudFront 事件执行函数时，发布另一个版本并编辑该函数以添加触发器。有关更多信息，请参阅 为 Lambda@Edge 函数添加触发器。

通过 API 或 AWS CLI 来使用 Lambda@Edge

您还可以使用 Lambda 和 CloudFront API 操作，以编程方式设置 Lambda@Edge 函数和 CloudFront 触发器。有关更多信息，请参阅以下主题：

- AWS LambdaAPI 引用
- Amazon CloudFront API 参考
- 您也可以使用以下 AWS Command Line Interface（AWS CLI）命令：
  - Lambda create-function
  - CloudFront create-distribution

  - CloudFront create-distribution-with-tags
  - CloudFront update-distribution
- AWS SDK（请参阅 SDK 和工具包部分。）
- AWS Tools for PowerShell Cmdlet 引用

更改 Lambda 函数

在创建 Lambda@Edge 函数后，可以使用 Lambda 控制台对其进行更改。

> **注意**
>
> - 原始版本标记为 $LATEST。
> - 您只能编辑 $LATEST 版本。
> - 每次编辑 $LATEST 版本时，均必须发布带编号的新版本。
> - 您无法为 $LATEST 创建触发器。
> - 当您发布函数的新版本时，Lambda 不会将触发器从以前的版本自动复制到新版本中。您必须为新版本重现触发器。
> - 当您将 CloudFront 事件的触发器添加到函数中时，如果已经有一个针对相同分配、缓存行为和同一函数早期版本的事件的触发器，则 Lambda 会从早期版本中删除该触发器。
> - 在更新 CloudFront 分配（如添加触发器）后，您必须等待更改传播到边缘站点，您在触发器中指定的函数才能运行。

更改 Lambda 函数（控制台）

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 在页面顶部的“区域”列表中，选择美国东部（弗吉尼亚州北部）。
3. 在函数列表中，选择函数的名称。

   默认情况下，控制台会显示 $LATEST 版本。您可以查看早期版本（选择 Qualifiers），但是只能编辑 $LATEST。
4. 在代码选项卡上，对于代码输入种类，选择在浏览器中编辑代码、上传 .zip 文件，或从 Amazon S3 上传文件。

5. 选择保存或保存并测试。
6. 选择操作，然后选择发布新版本。
7. 在 Publish new version from $LATEST 对话框中，输入新版本的描述。此描述会与自动生成的版本号一起显示在版本列表中。
8. 选择 Publish。

   新版本将自动成为最新版本。版本号会显示在页面左上角的版本中。
9. 选择触发器选项卡。
10. 选择 Add trigger。
11. 在添加触发器对话框中，选择虚线框，然后再选择 CloudFront。

> **Note**
>
> 如果您已为函数创建一个或多个触发器，则 CloudFront 为默认服务。

12. 指定以下值，以指示您希望 Lambda 函数何时执行。
    a. 分配 ID – 选择要向其中添加触发器的分配的 ID。
    b. 缓存行为 – 选择缓存行为，该行为将指定您要对其执行函数的对象。
    c. CloudFront 事件 – 选择促使函数执行的 CloudFront 事件。
    d. 启用触发器并复制 – 选中此复选框，以便 Lambda 将函数复制到全球各地的 AWS 区域。
13. 选择 Submit。
14. 要为该函数添加更多触发器，请重复步骤 10 到 13。

为 Lambda@Edge 函数添加触发器

Lambda@Edge 触发器是指 CloudFront 分配、缓存行为与使函数开始执行的事件的组合。您可以指定使函数运行的一个或多个 CloudFront 触发器。例如，您可以创建一个触发器，在 CloudFront 收到来自为分配所设置的特定缓存行为查看器的请求时执行函数。

> **Tip**
>
> 在创建 CloudFront 分配时，您可以指定一些设置来告诉 CloudFront 在收到不同的请求时如何响应。默认设置称为分配的默认缓存行为。您可以设置其他缓存行为来定义 CloudFront 在特定

情况下（例如，在收到特定文件类型的请求时）如何响应。有关更多信息，请参阅缓存行为设置。

在创建 Lambda 函数时，可以仅指定一个触发器。您可以通过使用 Lambda 控制台或在 CloudFront 控制台中编辑分配，在稍后向同一函数中添加更多触发器。

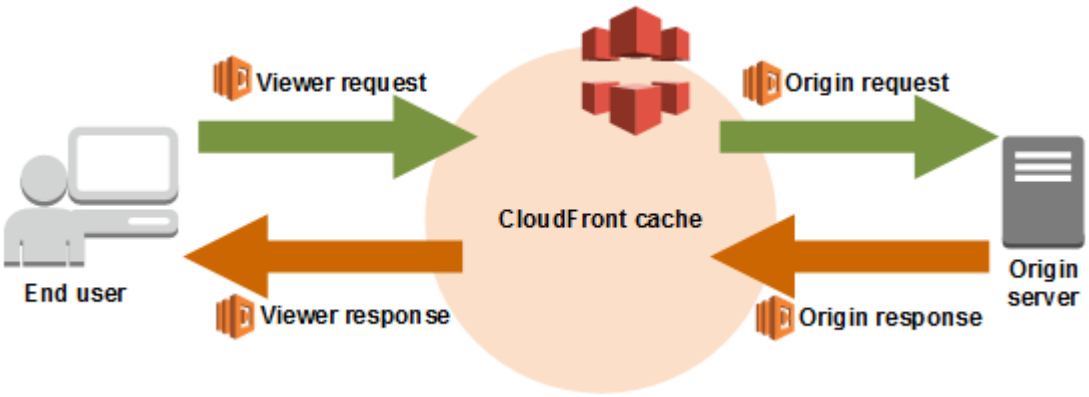
- 如果您要将更多触发器添加到同一 CloudFront 分配的函数中，那么使用 Lambda 控制台非常有效。
- 如果要为多个分配添加触发器，那么使用 CloudFront 控制台非常有效，因为这样更方便查找您要更新的分配。您同时还可以更新其他 CloudFront 设置。

> **Note**
>
> 要以编程方式使用 Lambda@Edge，请参阅通过 API 或 AWS CLI 来使用 Lambda@Edge。

主题

可以触发 Lambda@Edge 函数的 CloudFront 事件

对于 Amazon CloudFront 分配中的每个缓存行为，您最多可添加四个触发器（关联），以便在发生特定 CloudFront 事件时触发 Lambda 函数执行。CloudFront 触发器可以基于四个 CloudFront 事件之一，如下图所示。



可用于触发 Lambda@Edge 函数的 CloudFront 事件如下所示：

查看器请求

当 CloudFront 收到查看器的请求时及它检查请求的对象是否在 CloudFront 缓存中之前，该函数会执行。

源请求

仅当 CloudFront 将请求转发给您的源时，该函数才会执行。当请求的对象位于 CloudFront 缓存中时，该函数不会执行。

源响应

在 CloudFront 收到来自源的响应之后及它将对象缓存在响应中之前，该函数会执行。请注意，即使从源返回了错误，该函数仍会执行。

在以下情况下该函数不会执行：

- 当请求的文件位于 CloudFront 缓存中并且未过期时。
- 当从由源请求事件触发的函数中生成响应时。

查看器响应

在将请求的文件返回到查看器之前，该函数会执行。请注意，无论文件是否已在 CloudFront 缓存中，该函数都会执行。

在以下情况下该函数不会执行：

- 当源返回 400 或更高的 HTTP 状态代码时。
- 当返回自定义错误页面时。
- 当从由查看器请求事件触发的函数中生成响应时。
- 当 CloudFront 将 HTTP 请求自动重定向到 HTTPS 时（当 查看器协议策略 的值为将 HTTP 重定向到 HTTPS 时）。

当对同一个缓存行为添加多个触发器时，您可以使用它们运行同一个函数或对每个触发器运行不同的函数。也可以将同一个函数与多个分配关联。

> **Note**
>
> 当 CloudFront 事件触发 Lambda 函数的执行时，该函数必须完成，然后 CloudFront 才能继续。例如，如果 Lambda 函数被某个 CloudFront 查看器请求事件触发，则在 Lambda 函数完成运行之前，CloudFront 不会将响应返回给查看器或将请求转发到源。这意味着触发 Lambda 函数的每个请求均会增加请求的延迟，因此您可能希望该函数尽快执行。

决定使用哪个 CloudFront 事件来触发 Lambda@Edge 函数

当您决定使用哪个 CloudFront 事件来触发 Lambda 函数时，请考虑以下因素：

是否希望 CloudFront 缓存由 Lambda 函数更改的对象？

如果您希望 CloudFront 缓存 Lambda 函数修改的对象，以便在下次请求该对象时 CloudFront 可以从边缘站点中提供该对象，请使用源请求或源响应事件。这样可减少源上的负载、减少后续请求的延迟，并降低对后续请求调用 Lambda@Edge 的成本。

例如，如果要添加、删除或更改由源返回的对象的标头，并且希望 CloudFront 缓存结果，请使用源响应事件。

是否希望该函数针对每个请求执行？

如果您希望该函数针对 CloudFront 为分配接收的每个请求执行，请使用查看器请求或查看器响应事件。只有在未将请求的对象缓存在边缘站点中且 CloudFront 将请求转发到源时，才会发生源请求和源响应事件。

该函数是否会更改缓存键？

如果您希望该函数更改您要用作缓存基础的值，请使用查看器请求事件。例如，如果某个函数将 URL 更改为在路径中包含语言缩写 (例如，由于用户从下拉列表中选择了其语言)，请使用查看器请求事件：

- 查看器请求中的 URL – https://example.com/en/index.html
- 在请求来自德国的一个 IP 地址时的 URL – https://example.com/de/index.html

如果您要根据 Cookie 或请求标头缓存，则也使用查看器请求事件。

> **Note**
>
> 如果该函数更改 Cookie 或标头，则将 CloudFront 配置为将请求的适用部分转发到源。有关更多信息，请参阅以下主题：
>
> - 根据 Cookie 缓存内容
> - 根据请求标头缓存内容

该函数是否会影响来自源的响应？

如果希望该函数以会影响来自源的响应的方式更改请求，请使用源请求事件。通常，大多数查看器请求事件都不会被转发到源；CloudFront 使用已在边缘缓存中的对象来响应请求。如果该函数基于源请求事件更改请求，则 CloudFront 会缓存对更改的源请求的响应。

将触发器添加到 Lambda@Edge 函数中

您可以使用 AWS Lambda 控制台或 Amazon CloudFront 控制台向 Lambda@Edge 函数中添加触发器。

> **Important**
>
> 您只能为函数的编号版本（不能为 $LATEST）创建触发器。

Lambda console

将触发器添加到 Lambda@Edge 函数中

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。
2. 在页面顶部的“区域”列表中，选择美国东部（弗吉尼亚州北部）。
3. 在函数页面上，选择您要为其添加触发器的函数的名称。
4. 在函数概述页面上，选择版本选项卡。
5. 选择您要为其添加触发器的版本。

   选择某个版本后，相应按钮的名称会更改为 Version: $LATEST 或 Version: 版本号。
6. 选择触发器选项卡。
7. 选择添加触发器。
8. 在触发器配置中，选择选择源，输入 **`cloudfront`**，然后选择 CloudFront。

   > **Note**
   >
   > 如果您已创建一个或多个触发器，则 CloudFront 为默认服务。
9. 指定以下值，以指示您希望 Lambda 函数何时执行。

a. 分配 – 选择要向其中添加触发器的分配。

b. 缓存行为 – 选择缓存行为，该行为将指定您要对其执行函数的对象。

> **Note**
>
> 如果您对缓存行为指定 *，则 Lambda 函数会部署到默认缓存行为。

c. CloudFront 事件 – 选择促使函数执行的 CloudFront 事件。

d. 包括正文 – 如果要在函数中访问请求正文，请选中该复选框。

e. 确认部署到 Lambda@Edge – 选中该复选框，以便 AWS Lambda 将函数复制到全球各地的 AWS 区域。

10. 选择添加。

在更新的 CloudFront 分配部署后，函数开始处理指定 CloudFront 事件的请求。要确定是否已部署分配，请在导航窗格中选择分配。在部署分配后，分配的状态列的值将从正在部署更改为部署日期和时间。

CloudFront console

将 CloudFront 事件的触发器添加到 Lambda 函数

1. 获取您要为其添加触发器的 Lambda 函数的 ARN：

   a. 登录到 AWS Management Console，然后通过以下网址打开 AWS Lambda 控制台：https://console.aws.amazon.com/lambda/。

   b. 在页面顶部的区域列表中，选择美国东部（弗吉尼亚州北部）。

   c. 在函数列表中，选择您要为其添加触发器的函数的名称。

   d. 在函数概述页面上，选择版本选项卡，然后再选择要为其添加触发器的带编号的版本。

   e. 选择复制 ARN 按钮，将 ARN 复制到剪贴板。Lambda 函数的 ARN 如下所示：

   ```
   arn:aws:lambda:us-east-1:123456789012:function:TestFunction:2
   ```

   末尾的号码（在本示例中为 2）是函数的版本号。

2. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台

3. 在分配列表中，选择要为其添加触发器的分配的 ID。

4. 选择行为选项卡。

5. 选择要为其添加触发器的缓存行为，然后选择编辑。
6. 对于函数关联，请在函数类型列表中，选择 Lambda@Edge，以确定您希望何时执行函数：针对查看器请求、查看器响应、源请求或源响应。

   有关更多信息，请参阅 决定使用哪个 CloudFront 事件来触发 Lambda@Edge 函数。
7. 在函数 ARN/名称文本框中，粘贴当所选事件发生时您要执行的 Lambda 函数的 ARN。这是您从 Lambda 控制台复制的值。
8. 如果要在函数中访问请求正文，请选择包含正文。

   如果您仅希望替换请求正文，则不需要选择该选项。
9. 要对更多事件类型执行同一函数，请重复步骤 6 和 7。
10. 选择保存更改。
11. 要针对该分配为更多缓存行为添加触发器，请重复步骤 5 到 10。

    在更新的 CloudFront 分配部署后，函数开始处理指定 CloudFront 事件的请求。要确定是否已部署分配，请在导航窗格中选择分配。在部署分配后，分配的状态列的值将从正在部署更改为部署时间和日期。

测试和调试 Lambda@Edge 函数

本主题包括介绍测试和调试 Lambda@Edge 函数的策略的部分。请务必单独测试 Lambda@Edge 函数代码以确保它完成预期的任务，并进行集成测试以确保该函数在 CloudFront 中正常工作。

在集成测试期间或在部署函数后，您可能需要调试 CloudFront 错误，例如，HTTP 5xx 错误。错误可能是从 Lambda 函数返回的无效响应、触发该函数时的执行错误，或者由于 Lambda 服务限制执行而产生的错误。本主题中的部分使用相同的策略以确定问题是哪种故障类型，并采取相同的措施以纠正该问题。

> **Note**
>
> 如果在纠正错误时查看 CloudWatch 日志文件或指标，请注意它们显示或存储在位置最接近执行函数的 AWS 区域。因此，如果您的网站或 Web 应用程序用户位于英国，并且 Lambda 函数与您的分配关联，您必须更改区域以查看伦敦 AWS 区域的 CloudWatch 指标或日志文件。有关更多信息，请参阅 the section called “ 确定 Lambda@Edge 区域”。

主题

- 测试 Lambda@Edge 函数
- 识别 CloudFront 中的 Lambda@Edge 函数错误
- 排查 Lambda@Edge 函数响应无效问题（验证错误）
- 排查 Lambda@Edge 函数执行错误
- 确定 Lambda@Edge 区域
- 确定您的账户是否将日志推送到 CloudWatch

测试 Lambda@Edge 函数

可以使用两个步骤测试 Lambda 函数：单独测试和集成测试。

测试单独功能

在将 Lambda 函数添加到 CloudFront 之前，请确保先使用 Lambda 控制台中的测试功能或其他方法测试该功能。有关在 Lambda 控制台中进行测试的更多信息，请参阅《AWS Lambda 开发人员指南》中的使用控制台创建 Lambda 函数中的调用 Lambda 函数和验证结果、日志和指标部分。

在 CloudFront 中测试您的函数的运行情况

请务必完成集成测试，其中，您的函数与一个分配关联并根据 CloudFront 事件运行。确保为正确的事件触发函数，并为 CloudFront 返回有效且正确的响应。例如，确保事件结构正确无误，仅包含有效标头等等。

在 Lambda 控制台上重复对函数进行集成测试时，请在修改代码或更改调用函数的 CloudFront 触发器时参阅 Lambda@Edge 教程中的步骤。例如，确保您使用带编号的函数版本，如本教程中的以下步骤所述：第 4 步：添加 CloudFront 触发器来运行函数。

在进行更改和部署时，请注意需要几分钟的时间以在所有区域中复制更新的函数和 CloudFront 触发器。这通常需要几分钟，但最长需要 15 分钟。

您可以转到 CloudFront 控制台并查看您的分配，确认复制是否完成。

检查您的复制是否已完成部署

1. 通过 https://console.aws.amazon.com/cloudfront/v4/home 打开 CloudFront 控制台
2. 选择分配名称。
3. 查看分配的状态是否已从正在进行恢复为已部署，这意味着已复制函数。然后，按照下一节中的步骤验证函数是否正常工作。

请注意，控制台中的测试只会验证函数的逻辑，而不会应用特定于 Lambda@Edge 的任何服务限额（以前称为限制）。

识别 CloudFront 中的 Lambda@Edge 函数错误

在确认您的函数逻辑正常工作后，在 CloudFront 中运行函数时，您可能仍会看到 HTTP 5xx 错误。可能返回 HTTP 5xx 错误的原因有很多，其中包括 Lambda 函数错误或 CloudFront 中的其他问题。

- 如果您使用 Lambda@Edge 函数，则可以在 CloudFront 控制台中使用图表帮助跟踪导致错误的原因，然后进行修复。例如，您可以查看是 CloudFront 还是 Lambda 函数导致了 HTTP 5xx 错误，然后，对于特定函数，您可以查看相关的日志文件来调查问题。
- 要对 CloudFront 中的常规 HTTP 错误进行问题排查，请参阅以下主题中的问题排查步骤：对来自源的错误响应进行故障排除。

是什么原因导致 CloudFront 中的 Lambda@Edge 函数错误

Lambda 函数导致 HTTP 5xx 错误可能有很多原因，您应采取的故障排除措施取决于错误类型。错误可以归类如下：

Lambda 函数执行错误。

如果由于函数中存在未处理的异常或代码中存在错误，使得 CloudFront 没有从 Lambda 中收到响应，则会导致执行错误。例如，如果代码包含回调（错误）。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 函数错误。

向 CloudFront 返回无效的 Lambda 函数响应

函数运行后，CloudFront 会收到来自 Lambda 的响应。如果响应的对象结构不符合 Lambda@Edge 事件结构，或响应包含无效的标头或其他无效的字段，则会返回错误。

由于 Lambda 服务配额（以前称为限制），CloudFront 中的执行受到限制

Lambda 服务限制每个区域中的执行次数，并在超过配额时返回错误。

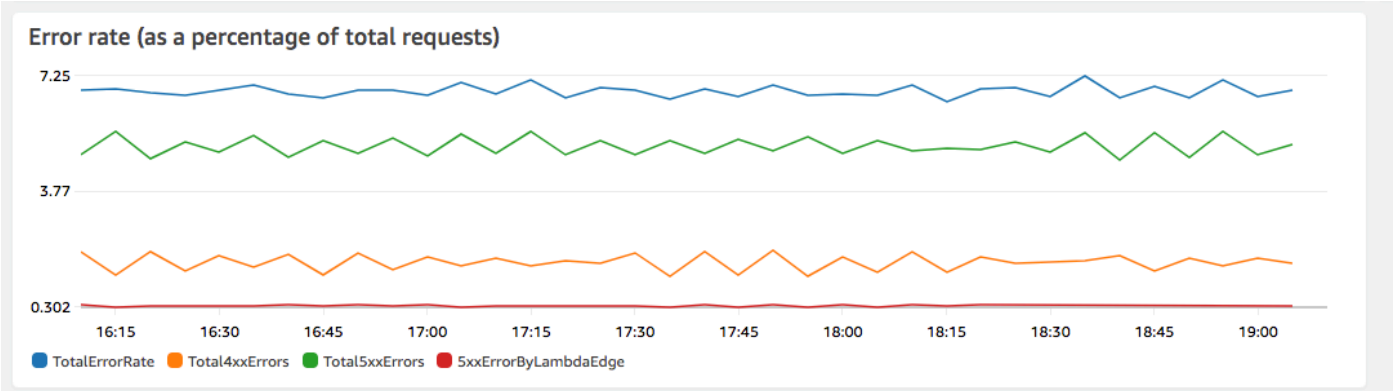
如何确定故障类型

为了帮助您在调试和处理以解决 CloudFront 返回的错误时确定重点，确定 CloudFront 返回 HTTP 错误的原因会有所帮助。要开始使用，您可以使用 AWS Management Console 上 CloudFront 控制台的监控部分中提供的图表。有关在 CloudFront 控制台的监控部分中查看图表的更多信息，请参阅使用 Amazon CloudWatch 监控 CloudFront 指标。

在您希望跟踪错误是由源还是由 Lambda 函数返回时，以及在错误源自 Lambda 函数时需要缩小问题类型的范围时，下列图表会非常有用。

错误率图表

在概览选项卡上，您可以看到各个分配的一个图表是错误率图表。此图表显示相对于传入到您分配的请求总数，错误率的百分比。图表显示总错误率、4xx 错误总数、5xx 错误总数以及来自 Lambda 函数的 5xx 错误总数。根据错误类型和卷，您可以采取措施来调查和排查造成错误的原因。



- 如果您看到 Lambda 错误，可以通过查看函数返回的特定类型错误来进一步进行调查。Lambda@Edge 错误选项卡包含按类型分类函数错误的图表，帮助您确定特定函数的问题。
- 如果您看到 CloudFront 错误，可以进行故障排查和处理来修复原始错误，或者更改 CloudFront 配置。有关更多信息，请参阅 对来自源的错误响应进行故障排除。

执行错误和无效函数响应图表

Lambda@Edge 错误选项卡包含针对特定分配按类型对 Lambda@Edge 错误进行分类的图表。例如，一个图表按 AWS 区域显示所有执行错误。

要更轻松地解决问题，您可以通过按区域打开并检查特定函数的日志，来查找特定问题。

按区域查看特定函数的日志文件

1. 在 Lambda@Edge 错误选项卡上，在关联的 Lambda@Edge 函数下，选择函数名称，然后选择查看指标。
2. 接下来，在包含函数名称的页面上，在右上角选择查看函数日志，然后选择一个区域。

   例如，如果您在错误图表中看到美国西部（俄勒冈州）区域存在问题，请从下拉列表中选择该区域。这将打开 Amazon CloudWatch 控制台。
3. 在该区域的 CloudWatch 控制台中，在日志流下，选择一个日志流以查看该函数的事件。

此外，阅读本章中的下列部分，了解有关问题排查和修复错误的更多建议。

限制图表

Lambda@Edge 错误选项卡还包括一个限制图表。有时候，在您达到了区域并发限制（以前称为限制）时，Lambda 服务会按区域限制函数调用。如果您发现超出限制错误，则您的函数已达到 Lambda 服务对在区域中执行所施加的配额。有关更多信息（包括如何请求提高配额），请参阅有关 Lambda@Edge 的配额。

有关如何在排查 HTTP 错误时使用此信息的示例，请参阅调试 AWS 上的内容分发的四个步骤。

排查 Lambda@Edge 函数响应无效问题（验证错误）

如果您发现问题是 Lambda 验证错误，则表示您的 Lambda 函数向 CloudFront 返回无效的响应。请按照此部分中的指导采取措施以检查您的函数，并确保您的响应符合 CloudFront 要求。

CloudFront 通过两种方式验证来自 Lambda 函数的响应：

- Lambda 响应必须符合所需的对象结构。不正确的对象结构示例包括：无法解析 JSON，缺少必填字段以及在响应中包含无效的对象。有关更多信息，请参阅 Lambda@Edge 事件结构。
- 响应只能包含有效的对象值。如果响应包含有效的对象，但具有不支持的值，则会出现错误。示例包括：添加或更新列入黑名单的标头或只读标头（请参阅 边缘函数的限制），超出最大正文大小（请参阅 Lambda 错误 主题中的生成的响应大小的限制）和无效的字符或值（请参阅 Lambda@Edge 事件结构）。

在 Lambda 向 CloudFront 返回无效的响应时，将在日志文件中写入错误消息，CloudFront 将这些日志文件推送到执行 Lambda 函数所在的区域中的 CloudWatch。这是在具有无效的响应时将日志文件发送

到 CloudWatch 的默认行为。不过，如果在发布该功能之前将 Lambda 函数与 CloudFront 相关联，则可能不会为您的函数启用该功能。有关更多信息，请参阅本主题后面的确定您的账户是否将日志推送到 CloudWatch。

CloudFront 将日志文件推送到与您函数的执行位置对应的区域（在与您的分配关联的日志组中）。日志组具有以下格式：`/aws/cloudfront/LambdaEdge/DistributionId`，其中 *`DistributionId`* 是您的分配的 ID。要确定可以找到 CloudWatch 日志文件的区域，请参阅本主题后面的确定 Lambda@Edge 区域。

如果可再现该错误，您可以创建一个导致该错误的新请求，然后在失败的 CloudFront 响应（`X-Amz-Cf-Id` 标头）中找到请求 ID 以在日志文件中找到单个故障。日志文件条目包含可帮助您确定返回错误的原因的信息，并且还会列出相应的 Lambda 请求 ID，以便您可以在单个请求的上下文中分析根本原因。

如果错误是间歇性的，您可以使用 CloudFront 访问日志查找失败请求的请求 ID，然后在 CloudWatch 日志中搜索相应的错误消息。有关更多信息，请参阅上一节确定故障类型。

排查 Lambda@Edge 函数执行错误

如果问题是 Lambda 执行错误，为 Lambda 函数创建日志记录语句以将消息写入到 CloudWatch 日志文件可能是非常有用的，从而在 CloudFront 中监视函数的执行情况并确定它是否正常工作。然后，您可以在 CloudWatch 日志文件中搜索这些语句，以验证您的函数是否正常工作。

> **Note**
>
> 即使您尚未更改 Lambda@Edge 函数，对该 Lambda 函数执行环境的更新也可能会影响它并可能返回执行错误。有关测试和迁移到更高版本的信息，请参阅对 AWS Lambda 和 AWS Lambda@Edge 执行环境的近期更新。

确定 Lambda@Edge 区域

要查看 Lambda@Edge 函数接收流量的区域，请在 AWS Management Console上的 CloudFront 控制台中查看此函数的指标。指标针对各个 AWS 区域显示。在同一页上，您可以选择一个区域并查看该区域的日志文件，从而调查问题。您必须查看相应 AWS 区域中的 CloudWatch 日志文件，以查看在 CloudFront 执行 Lambda 函数时创建的日志文件。

有关在 CloudFront 控制台的监控部分中查看图表的更多信息，请参阅使用 Amazon CloudWatch 监控 CloudFront 指标。

确定您的账户是否将日志推送到 CloudWatch

默认情况下，CloudFront 为无效的 Lambda 函数响应启用日志记录，并使用Lambda@Edge 的服务相关角色之一将日志文件推送到 CloudWatch。如果在发布无效的 Lambda 函数响应日志功能之前将 Lambda@Edge 函数添加到 CloudFront，下次更新 Lambda@Edge 配置时，将启用日志记录，例如，通过添加 CloudFront 触发器。

您可以执行以下操作，以验证是否为您的账户启用将日志文件推送到 CloudWatch 的功能：

- 检查以确定日志是否显示在 CloudWatch 中。确保您查看执行 Lambda@Edge 函数的区域。有关更多信息，请参阅 确定 Lambda@Edge 区域。
- 在 IAM 中确定在您的账户中是否存在相关的服务相关角色。为此，请打开 IAM 控制台 (https://console.aws.amazon.com/iam/)，然后选择角色以查看您的账户的服务相关角色列表。查找以下角色：`AWSServiceRoleForCloudFrontLogger`。

删除 Lambda@Edge 函数和副本

仅当 CloudFront 已创建 Lambda@Edge 函数的副本时，您才能删除该函数。在以下情况下，Lambda 函数的副本将自动删除：

- 在您从所有 CloudFront 分配中删除该函数的上一个关联后。如果多个分配使用一个函数，则仅在从上一个分配中删除函数关联后删除副本。
- 在您删除与函数关联的上一个分配后。

通常，将在数小时内删除副本。无法手动删除 Lambda@Edge 函数副本。这有助于防止出现删除仍在使用的副本的情况，这种情况将导致错误。

> **Warning**
>
> 不要构建使用 CloudFront 外部的 Lambda@Edge 函数副本的应用程序。当删除它们与分配的关联，或者删除分配本身时，将删除这些副本。可能在不发出警告的情况下删除外部应用程序所依赖的副本，这会导致其失败。

从 CloudFront 分配中删除 Lambda@Edge 函数关联（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。

2. 选择具有要删除的 Lambda@Edge 函数关联的分配的 ID。
3. 选择行为选项卡。
4. 选择具有要删除的 Lambda@Edge 函数关联的缓存行为，然后选择编辑。
5. 在函数关联的函数类型下，选择无关联以删除 Lambda@Edge 函数关联。
6. 选择保存更改。

从 CloudFront 分配中删除 Lambda@Edge 函数关联后，可以选择性地从 AWS Lambda 中删除 Lambda 函数或函数版本。删除函数关联后，等待几个小时，以便清理 Lambda@Edge 函数副本。之后，您将能够使用 Lambda 控制台、AWS、Lambda API 或 AWS CLI SDK 删除该函数。

您还可以删除特定版本的 Lambda 函数，前提是该版本没有任何与之关联的 CloudFront 分配。删除某个 Lambda 函数版本的所有关联后，请等待几小时。然后，您将能够删除该函数版本。

Lambda@Edge 事件结构

下面的主题介绍了触发 Lambda@Edge 函数时，CloudFront 传递给该函数的请求和响应事件对象。

主题

动态源选择

您可以在缓存行为中使用路径模式，根据所请求对象的路径和名称将请求路由到源，例如 `images/*.jpg`。使用 Lambda@Edge，您也可以基于其他功能将请求路由到源，例如请求标头中的值。

在多种情况下，这种动态源选择会非常有用。例如，您可以跨不同地理区域中的源分配请求，帮助实现全球负载均衡。或者，您可以选择性地将请求路由到不同的源，每个服务器提供特定功能：自动程序处理、SEO 优化、身份验证等。有关演示如何使用此功能的代码示例，请参阅 基于内容的动态源选择 - 示例。

在 CloudFront 源请求事件中，根据路径模式，事件结构中的 `origin` 对象包含有关将请求路由到的源的信息。您可以更新 `origin` 对象中的值，将请求路由到不同的源。更新 `origin` 对象时，您不需要在分配中定义源。您还可以将 Amazon S3 源对象替换为自定义源对象，或者执行相反的操作。但是，只能为每个请求指定一个源；该源可以是自定义源，也可以是 Amazon S3 源，但不能同时指定这两种源。

请求事件

下面的主题显示了 CloudFront 传递给查看器和源请求事件的 Lambda 函数的对象结构。这些示例显示了不带正文的 GET 请求。下面一些示例中，列出了查看器和源请求事件中的所有的可能字段。

主题

- 示例查看器请求
- 示例源请求
- 请求事件字段

示例查看器请求

下面的示例显示查看器请求事件对象。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "viewer-request",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "host": [
              {
                "key": "Host",
                "value": "d111111abcdef8.cloudfront.net"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "curl/7.66.0"
              }
            ],
            "accept": [
              {
```

```
                "key": "accept",
                "value": "*/*"
              }
            ]
          },
          "method": "GET",
          "querystring": "",
          "uri": "/"
        }
      }
    }
  ]
}
```

示例源请求

下面的示例显示源请求事件对象。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "origin-request",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "x-forwarded-for": [
              {
                "key": "X-Forwarded-For",
                "value": "203.0.113.178"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
                "value": "Amazon CloudFront"
              }
            ],
            "via": [
```

```
              {
                "key": "Via",
                "value": "2.0 2afae0d44e2540f472c0635ab62c232b.cloudfront.net
 (CloudFront)"
              }
            ],
            "host": [
              {
                "key": "Host",
                "value": "example.org"
              }
            ],
            "cache-control": [
              {
                "key": "Cache-Control",
                "value": "no-cache"
              }
            ]
          },
          "method": "GET",
          "origin": {
            "custom": {
              "customHeaders": {},
              "domainName": "example.org",
              "keepaliveTimeout": 5,
              "path": "",
              "port": 443,
              "protocol": "https",
              "readTimeout": 30,
              "sslProtocols": [
                "TLSv1",
                "TLSv1.1",
                "TLSv1.2"
              ]
            }
          },
          "querystring": "",
          "uri": "/"
        }
      }
    }
  ]
}
```

请求事件字段

请求事件对象数据包含在两个子对象中：config (Records.cf.config) 和 request (Records.cf.request)。下面的列表描述了各个子对象的字段。

Config 对象中的字段

下面的列表介绍了 config 对象 (Records.cf.config) 中的字段。

distributionDomainName（只读）

与请求关联的分配的域名。

distributionID（只读）

与请求关联的分配的 ID。

eventType（只读）

与请求关联的触发器类型：viewer-request 或 origin-request。

requestId（只读）

一个加密字符串，唯一地标识查看器到 CloudFront 的请求。requestId 值还在 CloudFront 访问日志中显示为 x-edge-request-id。有关更多信息，请参阅 配置和使用标准日志（访问日志）和 标准日志文件字段：

请求对象中的字段

下面的列表介绍了 request 对象 (Records.cf.request) 中的字段。

clientIp（只读）

发出请求的查看器的 IP 地址。如果查看器使用 HTTP 代理或负载均衡器发送请求，则值为该代理或负载均衡器的 IP 地址。

标头（读/写）

请求中的标头。请注意以下几点：

- headers 对象中的键为标准 HTTP 请求标头名称的小写版本。使用小写键可为您提供对标头值的不区分大小写的访问权限。
- 每个标头对象（例如，headers["accept"] 或 headers["host"]）是一个键/值对数组。对于一个指定标头，数组为请求中的每个值包含一个键/值对。

- key 包含 HTTP 请求中显示的标头的名称，名称区分大小写；例如 Host、User-Agent、X-Forwarded-For 等等。
- value 包含 HTTP 请求中显示的标头值。
- 当您的 Lambda 函数添加或修改请求标头，并且您未包含标头 key 字段时，Lambda@Edge 会自动使用您提供的标头名称插入标头 key。无论您如何格式化标头名称，自动插入的标头键都将通过对每个部分使用首字母大写方式 [用连字符 (-) 分隔] 来格式化。

  例如，您可以不带标头键添加标头 key，如下所示：

```
"user-agent": [
  {
    "value": "ExampleCustomUserAgent/1.X.0"
  }
]
```

  在本示例中，Lambda@Edge 会自动插入 "key": "User-Agent"。

有关标头使用情况限制的信息，请参阅边缘函数的限制。

method（只读）

请求中的 HTTP 方法。

querystring（读/写）

请求中的查询字符串（如果有的话）。如果请求中不包括查询字符串，则事件对象仍包括带空值的 querystring。有关查询字符串的更多信息，请参阅根据查询字符串参数缓存内容。

uri（读/写）

所请求对象的相对路径。如果您的 Lambda 函数修改了 uri 值，请记住以下事项：

- 新的 uri 值必须以正斜杠 (/) 开头。
- 如果某个函数更改 uri 值，则这样会更改查看器请求的对象。
- 如果某个函数更改 uri 值，则这样不会 更改该请求或该请求发送到的源的缓存行为。

body（读/写）

HTTP 请求的正文。body 结构可以包含以下字段：

inputTruncated（只读）

一个布尔值标记，它指示 Lambda@Edge 是否截断正文。有关更多信息，请参阅 具有 Include Body（包含正文）选项的请求正文的限制。

action（读/写）

您打算对正文执行的操作。action 选项如下所示：

- read-only：这是默认值。在从 Lambda 函数返回响应时，如果 action 是只读的，Lambda@Edge 将忽略对 encoding 或 data 的任何更改。
- replace：如果要替换发送到源的正文，请指定该选项。

encoding（读/写）

正文的编码。在 Lambda@Edge 向 Lambda 函数公开正文时，它先将正文转换为 base64-encoding。如果您为 action 选择 replace 以替换正文，您可以选择使用 base64（默认编码）或 text 编码。如果将 encoding 指定为 base64，但正文不是有效的 base64，CloudFront 将返回错误。

data（读/写）

请求正文内容。

origin（读/写）（仅限原始事件）

要将请求发送到的源。origin 结构只能包含一个源，它可以是自定义源或 Amazon S3 源。源结构可以包含以下字段：

customHeaders（读/写）（自定义和 Amazon S3 源）

您可以通过为每个自定义标头指定标头名称/值对，在请求中包括自定义标头。您不能添加不允许使用的标头，并且 Records.cf.request.headers 中不能存在同名标头。有关请求标头的注释也适用于自定义标头。有关更多信息，请参阅 CloudFront 无法添加到源请求的自定义标头和 边缘函数的限制：

domainName（读/写）（自定义和 Amazon S3 源）

源的域名。域名不能为空。

- 对于自定义源 – 指定 DNS 域名，例如 www.example.com。域名不能包含冒号 (:)，也不能为 IP 地址。域名最多可以有 253 个字符。
- 对于 Amazon S3 源 – 指定 Amazon S3 存储桶的 DNS 域名，例如 awsexamplebucket.s3.eu-west-1.amazonaws.com。名称必须最多为 128 个字符，并且必须为全小写。

path（读/写）（自定义和 Amazon S3 源）

源上的目录路径，请求应在其中查找内容。路径应该以正斜杠 (/) 开头，但不应该以正斜杠结尾（例如，它不应该以 example-path/ 结尾）。仅对于自定义源，路径应为 URL 编码，最大长度为 255 个字符。

keepaliveTimeout（读/写）（仅自定义源）

CloudFront 在接收最后一个响应数据包后应尝试与源保持连接的秒数。该值必须是 1 到 60 之间（含）的数字。

port（读/写）（仅自定义源）

自定义源中 CloudFront 应连接到的端口。端口必须为 80、443，或者是 1024 到 65535 之间（含）的数字。

protocol（读/写）（仅自定义源）

连接到您的源时 CloudFront 应使用的连接协议。该值可以是 `http` 或 `https`。

readTimeout（读/写）（仅自定义源）

向您的源发送请求后 CloudFront 应等待响应多长时间，以秒为单位。这还指定在 CloudFront 接收响应数据包之后应等待多长时间，然后再接收下一个数据包。该值必须是 4 到 60 之间（含）的数字。

如果您的使用案例需要的时间超过 60 秒，则可以为 `Response timeout per origin` 请求更高的配额。有关更多信息，请参阅 分配的一般配额。

sslProtocols（读/写）（仅自定义源）

CloudFront 在建立与您的源的 HTTPS 连接时可使用的最小 SSL/TLS 协议。可以是以下值之一：`TLSv1.2`、`TLSv1.1`、`TLSv1` 或 `SSLv3`。

authMethod（读/写）（仅限 Amazon S3 源）

如果您使用源访问身份 (OAI)，请将此字段设置为 `origin-access-identity`。如果您没有使用 OAI，请将其设置为 `none`。将 `authMethod` 设置为 `origin-access-identity` 时有以下几点要求：

- 您必须指定 `region`（请参阅以下字段）。
- 将请求从一个 Amazon S3 源更改为另一个源时，必须使用相同的 OAI。
- 将请求从自定义源更改为 Amazon S3 源时，不能使用 OAI。

> **Note**
>
> 此字段不支持源访问控制（OAC）。

region（读/写）（仅限 Amazon S3 源）

您的 Amazon S3 存储桶的 AWS 区域。仅当您将 authMethod 设置为 origin-access-identity 时，此项才是必需的。

响应事件

下面的主题显示了 CloudFront 传递给查看器和源响应事件的 Lambda 函数的对象结构。下面的示例是查看器和源响应事件中所有可能字段的列表。

主题

- 示例源响应
- 示例查看器响应
- 响应事件字段

示例源响应

下面的示例显示源响应事件对象。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "origin-response",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "x-forwarded-for": [
              {
                "key": "X-Forwarded-For",
                "value": "203.0.113.178"
              }
            ],
            "user-agent": [
              {
                "key": "User-Agent",
```

```
                "value": "Amazon CloudFront"
              }
            ],
            "via": [
              {
                "key": "Via",
                "value": "2.0 8f22423015641505b8c857a37450d6c0.cloudfront.net
(CloudFront)"
              }
            ],
            "host": [
              {
                "key": "Host",
                "value": "example.org"
              }
            ],
            "cache-control": [
              {
                "key": "Cache-Control",
                "value": "no-cache"
              }
            ]
          },
          "method": "GET",
          "origin": {
            "custom": {
              "customHeaders": {},
              "domainName": "example.org",
              "keepaliveTimeout": 5,
              "path": "",
              "port": 443,
              "protocol": "https",
              "readTimeout": 30,
              "sslProtocols": [
                "TLSv1",
                "TLSv1.1",
                "TLSv1.2"
              ]
            }
          },
          "querystring": "",
          "uri": "/"
        },
        "response": {
```

```
          "headers": {
            "access-control-allow-credentials": [
              {
                "key": "Access-Control-Allow-Credentials",
                "value": "true"
              }
            ],
            "access-control-allow-origin": [
              {
                "key": "Access-Control-Allow-Origin",
                "value": "*"
              }
            ],
            "date": [
              {
                "key": "Date",
                "value": "Mon, 13 Jan 2020 20:12:38 GMT"
              }
            ],
            "referrer-policy": [
              {
                "key": "Referrer-Policy",
                "value": "no-referrer-when-downgrade"
              }
            ],
            "server": [
              {
                "key": "Server",
                "value": "ExampleCustomOriginServer"
              }
            ],
            "x-content-type-options": [
              {
                "key": "X-Content-Type-Options",
                "value": "nosniff"
              }
            ],
            "x-frame-options": [
              {
                "key": "X-Frame-Options",
                "value": "DENY"
              }
            ],
            "x-xss-protection": [
```

```
            {
              "key": "X-XSS-Protection",
              "value": "1; mode=block"
            }
          ],
          "content-type": [
            {
              "key": "Content-Type",
              "value": "text/html; charset=utf-8"
            }
          ],
          "content-length": [
            {
              "key": "Content-Length",
              "value": "9593"
            }
          ]
        },
        "status": "200",
        "statusDescription": "OK"
      }
    }
  }
 ]
}
```

示例查看器响应

下面的示例显示查看器响应事件对象。

```
{
  "Records": [
    {
      "cf": {
        "config": {
          "distributionDomainName": "d111111abcdef8.cloudfront.net",
          "distributionId": "EDFDVBD6EXAMPLE",
          "eventType": "viewer-response",
          "requestId": "4TyzHTaYWb1GX1qTfsHhEqV6HUDd_BzoBZnwfnvQc_1oF26ClkoUSEQ=="
        },
        "request": {
          "clientIp": "203.0.113.178",
          "headers": {
            "host": [
```

```
            {
              "key": "Host",
              "value": "d111111abcdef8.cloudfront.net"
            }
          ],
          "user-agent": [
            {
              "key": "User-Agent",
              "value": "curl/7.66.0"
            }
          ],
          "accept": [
            {
              "key": "accept",
              "value": "*/*"
            }
          ]
        },
        "method": "GET",
        "querystring": "",
        "uri": "/"
      },
      "response": {
        "headers": {
          "access-control-allow-credentials": [
            {
              "key": "Access-Control-Allow-Credentials",
              "value": "true"
            }
          ],
          "access-control-allow-origin": [
            {
              "key": "Access-Control-Allow-Origin",
              "value": "*"
            }
          ],
          "date": [
            {
              "key": "Date",
              "value": "Mon, 13 Jan 2020 20:14:56 GMT"
            }
          ],
          "referrer-policy": [
            {
```

```
            "key": "Referrer-Policy",
            "value": "no-referrer-when-downgrade"
          }
        ],
        "server": [
          {
            "key": "Server",
            "value": "ExampleCustomOriginServer"
          }
        ],
        "x-content-type-options": [
          {
            "key": "X-Content-Type-Options",
            "value": "nosniff"
          }
        ],
        "x-frame-options": [
          {
            "key": "X-Frame-Options",
            "value": "DENY"
          }
        ],
        "x-xss-protection": [
          {
            "key": "X-XSS-Protection",
            "value": "1; mode=block"
          }
        ],
        "age": [
          {
            "key": "Age",
            "value": "2402"
          }
        ],
        "content-type": [
          {
            "key": "Content-Type",
            "value": "text/html; charset=utf-8"
          }
        ],
        "content-length": [
          {
            "key": "Content-Length",
            "value": "9593"
```

```
              }
            ]
          },
          "status": "200",
          "statusDescription": "OK"
        }
      }
    }
  ]
}
```

响应事件字段

响应事件对象数据包含在三个子对象中：config (Records.cf.config)、request (Records.cf.request) 和 response (Records.cf.response)。有关请求对象中的字段的更多信息，请参阅请求对象中的字段。下面的列表描述了 config 和 response 子对象中的字段。

Config 对象中的字段

下面的列表介绍了 config 对象 (Records.cf.config) 中的字段。

distributionDomainName（只读）

与响应关联的分配的域名。

distributionID（只读）

与响应关联的分配的 ID。

eventType（只读）

与响应关联的触发器的类型：origin-response 或 viewer-response。

requestId（只读）

一个加密字符串，唯一地标识与此响应关联的查看器到 CloudFront 的请求。requestId 值还在 CloudFront 访问日志中显示为 x-edge-request-id。有关更多信息，请参阅 配置和使用标准日志（访问日志）和 标准日志文件字段：

响应对象中的字段

下面的列表介绍了 response 对象 (Records.cf.response) 中的字段。有关使用 Lambda@Edge 函数生成 HTTP 响应的信息，请参阅在请求触发器中生成 HTTP 响应。

headers（读/写）

响应中的标头。请注意以下几点：

- headers 对象中的键为标准 HTTP 请求标头名称的小写版本。使用小写键可为您提供对标头值的不区分大小写的访问权限。
- 每个标头对象（例如，headers["content-type"] 或 headers["content-length"]）是一个键/值对数组。对于一个指定标头，数组为响应中的每个值包含一个键/值对。
- key 包含 HTTP 响应中显示的标头的名称，名称区分大小写；例如 Content-Type、Content-Length、Cookie 等等。
- value 包含 HTTP 响应中显示的标头值。
- 当您的 Lambda 函数添加或修改响应标头，并且您不包含标头 key 字段时，Lambda@Edge 会自动使用您提供的标头名称插入标头 key。无论您如何格式化标头名称，自动插入的标头键都将通过对每个部分使用首字母大写方式 [用连字符 (-) 分隔] 来格式化。

  例如，您可以不带标头键添加标头 key，如下所示：

```
"content-type": [
  {
    "value": "text/html;charset=UTF-8"
  }
]
```

  在本示例中，Lambda@Edge 会自动插入 "key": "Content-Type"。

有关标头使用情况限制的信息，请参阅边缘函数的限制。

status

响应的 HTTP 状态代码。

statusDescription

响应的 HTTP 状态描述。

使用请求和响应

本节中的主题说明几种使用 Lambda@Edge 请求和响应的方法。

主题

- 将 Lambda@Edge 函数与源故障转移结合使用
- 在请求触发器中生成 HTTP 响应
- 更新源响应触发器中的 HTTP 响应
- 通过选择“包含正文”选项来访问请求正文

将 Lambda@Edge 函数与源故障转移结合使用

您可以将 Lambda@Edge 函数与您设置有源组的 CloudFront 分配结合使用，例如，对于您配置为帮助确保高可用性的源故障转移。要将 Lambda 函数与源组结合使用，请在创建缓存行为时，在源组的源请求或源响应触发器中指定此函数。

有关更多信息，请参阅下列内容：

- 创建源组：创建源组
- 源故障转移如何与 Lambda@Edge 结合使用：将源故障转移与 Lambda@Edge 函数结合使用

在请求触发器中生成 HTTP 响应

CloudFront 收到请求时，您可以使用 Lambda 函数生成 CloudFront 直接返回到查看器的 HTTP 响应，无需将响应转发到源。生成 HTTP 响应会减少源上的负载，通常也可以减少查看器的延迟。

生成 HTTP 响应的一些常见情况包括：

- 将小网页返回到查看器
- 返回 HTTP 301 或 302 状态代码，以便将用户重定向到其他网页
- 用户未通过身份验证时向查看器返回 HTTP 401 状态代码

出现以下 CloudFront 事件时，Lambda@Edge 函数可以生成 HTTP 响应：

查看器请求事件

当查看器请求事件触发了函数时，CloudFront 将响应返回到查看器并且不进行缓存。

源请求事件

当源请求事件触发函数时，CloudFront 检查边缘缓存中以前由函数生成的响应。

- 如果响应在缓存中，函数不执行，CloudFront 将缓存的响应返回查看器。
- 如果响应不在缓存中，则执行函数，CloudFront 将响应返回查看器，并且缓存它。

要查看生成 HTTP 响应的一些代码示例，请参阅 Lambda@Edge 函数示例。您也可以替换响应触发器中的 HTTP 响应。有关更多信息，请参阅 更新源响应触发器中的 HTTP 响应。

编程模型

本节介绍了使用 Lambda@Edge 生成 HTTP 响应的编程模型的信息。

主题

- 响应对象
- 错误
- 必填字段

响应对象

作为 `result` 方法的 `callback` 参数返回的响应应具有以下结构（请注意，只有 `status` 字段为必填字段）。

```
const response = {
    body: 'content',
    bodyEncoding: 'text' | 'base64',
    headers: {
        'header name in lowercase': [{
            key: 'header name in standard case',
            value: 'header value'
         }],
         ...
    },
    status: 'HTTP status code (string)',
    statusDescription: 'status description'
};
```

响应对象可包括以下值：

body

您希望 CloudFront 在生成的响应中返回的正文（如果有）。

bodyEncoding

您在 body 中指定的值的编码。有效编码仅为 `text` 和 `base64`。如果您在 `response` 对象中包含 body 但省略 `bodyEncoding`，CloudFront 将正文视为文本。

如果将 bodyEncoding 指定为 base64，但正文不是有效的 base64，CloudFront 将返回错误。

headers

您希望 CloudFront 在生成的响应中返回的标头。请注意以下几点：

- headers 对象中的键为标准 HTTP 请求标头名称的小写版本。使用小写键可为您提供对标头值的不区分大小写的访问权限。
- 每个标头（例如，headers["accept"] 或 headers["host"]）是一个键值对数组。对于一个指定标头，数组为生成的响应中的每个值包含一个键值对。
- key（可选）是显示在 HTTP 请求中的标头名称，区分大小写；例如 accept 或 host。
- 指定 value 作为标头值。
- 如果您不包括键值对的标头键部分，Lambda@Edge 会使用您提供的标头名称自动插入标头键。无论您如何格式化标头名称，所插入的标头键都会自动通过对每个部分使用首字母大写方式（用连字符 (-) 分隔）来格式化。

  例如，您可以不带标头键添加标头，如下所示：'content-type': [{ value: 'text/html;charset=UTF-8' }]

  在本示例中，Lambda@Edge 创建以下标头键：Content-Type。

有关标头使用情况限制的信息，请参阅边缘函数的限制。

status

HTTP 状态代码。以字符串形式提供状态代码。CloudFront 将提供的状态代码用于以下各项：

- 在响应中返回
- 当从由源请求事件触发的函数生成响应时，缓存在 CloudFront 边缘缓存中
- 登录 CloudFront 配置和使用标准日志（访问日志）

如果 status 值并非介于 200 到 599 之间，CloudFront 向查看器返回错误。

statusDescription

您希望 CloudFront 在响应中随 HTTP 状态代码一起返回的说明。无需使用标准说明，例如为 HTTP 状态代码 200 使用 OK。

错误

以下是所生成的 HTTP 响应可能的错误。

响应包含正文并为状态指定了 204（无内容）

当函数由查看器请求触发时，如果满足以下情况，则 CloudFront 向查看器返回 HTTP 502 状态代码（无效网关）：

- `status` 的值为 204（无内容）
- 响应包括 `body` 的值

这是因为 Lambda@Edge 会施加 RFC 2616 中介绍的可选限制，其中指出 HTTP 204 响应不需要包含消息正文。

对所生成响应的大小的限制

由 Lambda 函数生成的响应的最大大小取决于触发该函数的事件：

- 查看器请求事件 – 40 KB
- 源请求事件 – 1 MB

如果响应大于允许的大小，则 CloudFront 会向查看器返回 HTTP 502 状态代码（无效网关）。

必填字段

`status` 字段为必填项。

其他所有字段均为可选字段。

更新源响应触发器中的 HTTP 响应

当 CloudFront 从源服务器接收 HTTP 响应时，如果有源响应触发器与缓存行为关联，您可以修改 HTTP 响应以覆盖从源返回的内容。

更新 HTTP 响应的一些常见情况包括：

- 更改状态以设置 HTTP 200 状态代码并创建静态正文内容，这些内容在源服务器返回错误代码 (4xx 或 5xx) 时返回到查看器。有关代码示例，请参阅 示例：使用源响应触发器将错误状态代码更新为 200。
- 更改状态以设置 HTTP 301 或 HTTP 302 状态代码，用于在源返回错误状态代码 (4xx 或 5xx) 时将用户重定向到其他网站。有关代码示例，请参阅 示例：使用源响应触发器将错误状态代码更新为 302。

> **Note**
>
> 函数必须返回值介于 200 到 599 之间（包括这两者）的状态，否则 CloudFront 向查看器返回错误。

您也可以替换查看器和源请求事件中的 HTTP 响应。有关更多信息，请参阅 在请求触发器中生成 HTTP 响应。

当您在处理 HTTP 响应时，Lambda@Edge 不会将源服务器返回的正文公开到源响应触发器。您可以通过将其设置为所需值来生成静态内容正文，或者通过将值设置为空删除函数中的正文。如果您不更新函数中的正文字段，源服务器返回的源正文将返回到查看器。

通过选择“包含正文”选项来访问请求正文

您可以选择让 Lambda@Edge 为可写的 HTTP 方法（POST、PUT、DELETE 等）公开请求中的正文，以便您可以在 Lambda 函数中访问正文。您可以选择只读访问，也可以指定将替换正文。

要启用该选项，请在为函数创建 CloudFront 触发器以用于查看器请求或源请求事件时选择包含正文。有关更多信息，请参阅为 Lambda@Edge 函数添加触发器；要了解如何在函数中使用包含正文，请参阅Lambda@Edge 事件结构。

您可能希望使用此功能的情况包括：

- 处理 Web 表单（例如“联系我们”表单），而不将客户输入数据发送回源服务器。
- 收集查看器浏览器发送的 Web 信标数据并在边缘站点中进行处理。

有关代码示例，请参阅 Lambda@Edge 函数示例。

> **Note**
>
> 如果请求正文很大，Lambda@Edge 将会截断正文。有关最大大小和截断的详细信息，请参阅具有 Include Body（包含正文）选项的请求正文的限制。

Lambda@Edge 函数示例

有关将 Lambda 函数用于 Amazon CloudFront 的示例，请参阅以下各节。

> **Note**
>
> 如果您为 Lambda@Edge 函数选择运行时 Node.js 18 或更高版本，则会自动为您创建一个 `index.mjs` 文件。要使用以下代码示例，请改为将 `index.mjs` 文件重命名为 `index.js`。

主题

一般示例

本节中的示例说明一些在 CloudFront 中使用 Lambda@Edge 的常用方法。

主题

示例：A/B 测试

您可以使用以下示例测试图像的两个不同版本，不需要创建重定向或更改 URL。本示例读取查看器请求中的 Cookie，并相应地修改请求 URL。如果查看器未发送包含预期值之一的 Cookie，则示例会将查看器随机分配给其中一个 URL。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;
```

```
    if (request.uri !== '/experiment-pixel.jpg') {
        // do not process if this is not an A-B test request
        callback(null, request);
        return;
    }

    const cookieExperimentA = 'X-Experiment-Name=A';
    const cookieExperimentB = 'X-Experiment-Name=B';
    const pathExperimentA = '/experiment-group/control-pixel.jpg';
    const pathExperimentB = '/experiment-group/treatment-pixel.jpg';

    /*
     * Lambda at the Edge headers are array objects.
     *
     * Client may send multiple Cookie headers, i.e.:
     * > GET /viewerRes/test HTTP/1.1
     * > User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1
OpenSSL/1.0.1u zlib/1.2.3
     * > Cookie: First=1; Second=2
     * > Cookie: ClientCode=abc
     * > Host: example.com
     *
     * You can access the first Cookie header at headers["cookie"][0].value
     * and the second at headers["cookie"][1].value.
     *
     * Header values are not parsed. In the example above,
     * headers["cookie"][0].value is equal to "First=1; Second=2"
     */
    let experimentUri;
    if (headers.cookie) {
        for (let i = 0; i < headers.cookie.length; i++) {
            if (headers.cookie[i].value.indexOf(cookieExperimentA) >= 0) {
                console.log('Experiment A cookie found');
                experimentUri = pathExperimentA;
                break;
            } else if (headers.cookie[i].value.indexOf(cookieExperimentB) >= 0) {
                console.log('Experiment B cookie found');
                experimentUri = pathExperimentB;
                break;
            }
        }
    }

    if (!experimentUri) {
```

```
        console.log('Experiment cookie has not been found. Throwing dice...');
        if (Math.random() < 0.75) {
            experimentUri = pathExperimentA;
        } else {
            experimentUri = pathExperimentB;
        }
    }

    request.uri = experimentUri;
    console.log(`Request uri set to "${request.uri}"`);
    callback(null, request);
};
```

Python

```
import json
import random

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    if request['uri'] != '/experiment-pixel.jpg':
        # Not an A/B Test
        return request

    cookieExperimentA, cookieExperimentB = 'X-Experiment-Name=A', 'X-Experiment-
Name=B'
    pathExperimentA, pathExperimentB = '/experiment-group/control-pixel.jpg', '/
experiment-group/treatment-pixel.jpg'

    '''
    Lambda at the Edge headers are array objects.

    Client may send multiple cookie headers. For example:
    > GET /viewerRes/test HTTP/1.1
    > User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1
 OpenSSL/1.0.1u zlib/1.2.3
    > Cookie: First=1; Second=2
    > Cookie: ClientCode=abc
    > Host: example.com

    You can access the first Cookie header at headers["cookie"][0].value
```

```
    and the second at headers["cookie"][1].value.

    Header values are not parsed. In the example above,
    headers["cookie"][0].value is equal to "First=1; Second=2"
    '''

    experimentUri = ""

    for cookie in headers.get('cookie', []):
        if cookieExperimentA in cookie['value']:
            print("Experiment A cookie found")
            experimentUri = pathExperimentA
            break
        elif cookieExperimentB in cookie['value']:
            print("Experiment B cookie found")
            experimentUri = pathExperimentB
            break

    if not experimentUri:
        print("Experiment cookie has not been found. Throwing dice...")
        if random.random() < 0.75:
            experimentUri = pathExperimentA
        else:
            experimentUri = pathExperimentB

    request['uri'] = experimentUri
    print(f"Request uri set to {experimentUri}")
    return request
```

示例：覆盖响应标头

以下示例演示了如何基于其他标头的值来更改响应标头的值。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
   const response = event.Records[0].cf.response;
   const headers = response.headers;

   const headerNameSrc = 'X-Amz-Meta-Last-Modified';
   const headerNameDst = 'Last-Modified';
```

```
   if (headers[headerNameSrc.toLowerCase()]) {
      headers[headerNameDst.toLowerCase()] = [
         headers[headerNameSrc.toLowerCase()][0],
      ];
      console.log(`Response header "${headerNameDst}" was set to ` +
               `"${headers[headerNameDst.toLowerCase()][0].value}"`);
   }

   callback(null, response);
};
```

Python

```
import json

def lambda_handler(event, context):
    response = event["Records"][0]["cf"]["response"]
    headers = response["headers"]

    headerNameSrc = "X-Amz-Meta-Last-Modified"
    headerNameDst = "Last-Modified"

    if headers.get(headerNameSrc.lower(), None):
        headers[headerNameDst.lower()] = [headers[headerNameSrc.lower()][0]]
        print(f"Response header {headerNameDst.lower()} was set to
 {headers[headerNameSrc.lower()][0]}")

    return response
```

生成响应 - 示例

本节中的示例向您展示如何可以使用 Lambda@Edge 来生成响应。

主题

- 示例：提供静态内容（生成的响应）
- 示例：生成 HTTP 重定向（生成的响应）

示例：提供静态内容（生成的响应）

以下示例演示了如何使用 Lambda 函数来提供静态网站内容，这样可减少源服务器上的负载，并减少总体延迟。

> **Note**
>
> 您可以针对查看器请求和源请求事件生成 HTTP 响应。有关更多信息，请参阅 the section called “在请求触发器中生成 HTTP 响应”。
> 您也可以替换或删除源响应事件中 HTTP 响应的正文。有关更多信息，请参阅 the section called “更新源响应触发器中的 HTTP 响应”。

Node.js

```
'use strict';

const content = `
<\!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Simple Lambda@Edge Static Content Response</title>
  </head>
  <body>
    <p>Hello from Lambda@Edge!</p>
  </body>
</html>
`;

exports.handler = (event, context, callback) => {
    /*
     * Generate HTTP OK response using 200 status code with HTML body.
     */
    const response = {
        status: '200',
        statusDescription: 'OK',
        headers: {
            'cache-control': [{
                key: 'Cache-Control',
                value: 'max-age=100'
            }],
```

```
            'content-type': [{
                key: 'Content-Type',
                value: 'text/html'
            }]
        },
        body: content,
    };
    callback(null, response);
};
```

Python

```
import json

CONTENT = """
<\!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Simple Lambda@Edge Static Content Response</title>
</head>
<body>
    <p>Hello from Lambda@Edge!</p>
</body>
</html>
"""

def lambda_handler(event, context):
    # Generate HTTP OK response using 200 status code with HTML body.
    response = {
        'status': '200',
        'statusDescription': 'OK',
        'headers': {
            'cache-control': [
                {
                    'key': 'Cache-Control',
                    'value': 'max-age=100'
                }
            ],
            "content-type": [
                {
                    'key': 'Content-Type',
                    'value': 'text/html'
```

```
                }
            ]
        },
        'body': CONTENT
    }
    return response
```

示例：生成 HTTP 重定向（生成的响应）

以下示例演示了如何生成 HTTP 重定向。

> **Note**
>
> 您可以针对查看器请求和源请求事件生成 HTTP 响应。有关更多信息，请参阅 在请求触发器中生成 HTTP 响应。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    /*
     * Generate HTTP redirect response with 302 status code and Location header.
     */
    const response = {
        status: '302',
        statusDescription: 'Found',
        headers: {
            location: [{
                key: 'Location',
                value: 'https://docs.aws.amazon.com/lambda/latest/dg/lambda-
edge.html',
            }],
        },
    };
    callback(null, response);
};
```

Python

```
def lambda_handler(event, context):

    # Generate HTTP redirect response with 302 status code and Location header.

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': 'https://docs.aws.amazon.com/lambda/latest/dg/lambda-
edge.html'
            }]
        }
    }

    return response
```

查询字符串 - 示例

本节中的示例包括您可以将 Lambda@Edge 与查询字符串一起使用的方法。

主题

- 示例：根据查询字符串参数添加标头
- 示例：标准化查询字符串参数以提高缓存命中率
- 示例：将未经身份验证的用户重定向到登录页面

示例：根据查询字符串参数添加标头

下面的示例演示如何获取查询字符串参数的键-值对，然后根据这些值添加标头。

Node.js

```
'use strict';

const querystring = require('querystring');
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
```

```
    /* When a request contains a query string key-value pair but the origin server
     * expects the value in a header, you can use this Lambda function to
     * convert the key-value pair to a header. Here's what the function does:
     * 1. Parses the query string and gets the key-value pair.
     * 2. Adds a header to the request using the key-value pair that the function
 got in step 1.
     */

    /* Parse request querystring to get javascript object */
    const params = querystring.parse(request.querystring);

    /* Move auth param from querystring to headers */
    const headerName = 'Auth-Header';
    request.headers[headerName.toLowerCase()] = [{ key: headerName, value:
 params.auth }];
    delete params.auth;

    /* Update request querystring */
    request.querystring = querystring.stringify(params);

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    '''
    When a request contains a query string key-value pair but the origin server
    expects the value in a header, you can use this Lambda function to
    convert the key-value pair to a header. Here's what the function does:
        1. Parses the query string and gets the key-value pair.
        2. Adds a header to the request using the key-value pair that the function
 got in step 1.
    '''

    # Parse request querystring to get dictionary/json
    params = {k : v[0] for k, v in parse_qs(request['querystring']).items()}
```

```
    # Move auth param from querystring to headers
    headerName = 'Auth-Header'
    request['headers'][headerName.lower()] = [{'key': headerName, 'value':
 params['auth']}]
    del params['auth']

    # Update request querystring
    request['querystring'] = urlencode(params)

    return request
```

示例：标准化查询字符串参数以提高缓存命中率

下面的示例演示如何在 CloudFront 将请求转发给您的源之前通过对查询字符串进行以下更改来提高缓存命中率：

- 按参数名称的字母顺序排列键/值对
- 将键/值对的大小写更改为小写

有关更多信息，请参阅 根据查询字符串参数缓存内容。

Node.js

```
'use strict';

const querystring = require('querystring');

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    /* When you configure a distribution to forward query strings to the origin and
     * to cache based on an allowlist of query string parameters, we recommend
     * the following to improve the cache-hit ratio:
     * - Always list parameters in the same order.
     * - Use the same case for parameter names and values.
     *
     * This function normalizes query strings so that parameter names and values
     * are lowercase and parameter names are in alphabetical order.
     *
     * For more information, see:
     * https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/
QueryStringParameters.html
```

```
     */

    console.log('Query String: ', request.querystring);

    /* Parse request query string to get javascript object */
    const params = querystring.parse(request.querystring.toLowerCase());
    const sortedParams = {};

    /* Sort param keys */
    Object.keys(params).sort().forEach(key => {
        sortedParams[key] = params[key];
    });

    /* Update request querystring with normalized  */
    request.querystring = querystring.stringify(sortedParams);

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    '''
    When you configure a distribution to forward query strings to the origin and
    to cache based on an allowlist of query string parameters, we recommend
    the following to improve the cache-hit ratio:
    Always list parameters in the same order.
    - Use the same case for parameter names and values.

    This function normalizes query strings so that parameter names and values
    are lowercase and parameter names are in alphabetical order.

    For more information, see:
    https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/
QueryStringParameters.html
    '''
    print("Query string: ", request["querystring"])

    # Parse request query string to get js object
    params = {k : v[0] for k, v in parse_qs(request['querystring'].lower()).items()}
```

```
    # Sort param keys
    sortedParams = sorted(params.items(), key=lambda x: x[0])

    # Update request querystring with normalized
    request['querystring'] = urlencode(sortedParams)

    return request
```

示例：将未经身份验证的用户重定向到登录页面

下面的示例演示如何将未输入其凭证的用户重定向到登录页面。

Node.js

```
'use strict';

function parseCookies(headers) {
    const parsedCookie = {};
    if (headers.cookie) {
        headers.cookie[0].value.split(';').forEach((cookie) => {
            if (cookie) {
                const parts = cookie.split('=');
                parsedCookie[parts[0].trim()] = parts[1].trim();
            }
        });
    }
    return parsedCookie;
}

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /* Check for session-id in request cookie in viewer-request event,
     * if session-id is absent, redirect the user to sign in page with original
     * request sent as redirect_url in query params.
     */

    /* Check for session-id in cookie, if present then proceed with request */
    const parsedCookies = parseCookies(headers);
    if (parsedCookies && parsedCookies['session-id']) {
        callback(null, request);
```

```
        return;
    }

    /* URI encode the original request to be sent as redirect_url in query params */
    const encodedRedirectUrl = encodeURIComponent(`https://
${headers.host[0].value}${request.uri}?${request.querystring}`);
    const response = {
        status: '302',
        statusDescription: 'Found',
        headers: {
            location: [{
                key: 'Location',
                value: `https://www.example.com/signin?redirect_url=
${encodedRedirectUrl}`,
            }],
        },
    };
    callback(null, response);
};
```

Python

```
import urllib

def parseCookies(headers):
    parsedCookie = {}
    if headers.get('cookie'):
        for cookie in headers['cookie'][0]['value'].split(';'):
            if cookie:
                parts = cookie.split('=')
                parsedCookie[parts[0].strip()] = parts[1].strip()
    return parsedCookie

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Check for session-id in request cookie in viewer-request event,
    if session-id is absent, redirect the user to sign in page with original
    request sent as redirect_url in query params.
    '''
```

```
    # Check for session-id in cookie, if present, then proceed with request
    parsedCookies = parseCookies(headers)

    if parsedCookies and parsedCookies['session-id']:
        return request

    # URI encode the original request to be sent as redirect_url in query params
    redirectUrl = "https://%s%s?%s" % (headers['host'][0]['value'], request['uri'],
request['querystring'])
    encodedRedirectUrl = urllib.parse.quote_plus(redirectUrl.encode('utf-8'))

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': 'https://www.example.com/signin?redirect_url=%s' %
encodedRedirectUrl
            }]
        }
    }
    return response
```

按国家/地区或设备类型标头个性化内容 - 示例

本节中的示例说明您如何可以使用 Lambda@Edge 以基于位置或查看器使用的设备类型自定义行为。

主题

- 示例：将查看器请求重定向到国家/地区特定的 URL
- 示例：根据设备提供不同版本的对象

示例：将查看器请求重定向到国家/地区特定的 URL

下面的示例演示如何生成包含国家/地区特定的 URL 的 HTTP 重定向响应并将该响应返回到查看器。在您希望提供国家/地区特定的响应时，这非常有用。例如：

- 如果您有国家/地区特定的子域，例如 us.example.com 和 tw.example.com，则在查看器请求 example.com 时，您可以生成重定向响应。

- 如果您要流式传输视频，但您在特定国家/地区中无权流式传输内容，则可以将该国家/地区中的用户重定向到说明他们为何无法观看视频的页面。

请注意以下几点：

- 您必须将您的分配配置为基于 CloudFront-Viewer-Country 标头进行缓存。有关更多信息，请参阅 基于选择的请求标头进行缓存。
- CloudFront 在查看器请求事件之后添加 CloudFront-Viewer-Country 标头。要使用此示例，您必须为源请求事件创建触发器。

Node.js

```
'use strict';

/* This is an origin request function */
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /*
     * Based on the value of the CloudFront-Viewer-Country header, generate an
     * HTTP status code 302 (Redirect) response, and return a country-specific
     * URL in the Location header.
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Viewer-Country header. For more information, see
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
     *       2. CloudFront adds the CloudFront-Viewer-Country header after the
 viewer
     *          request event. To use this example, you must create a trigger for
 the
     *          origin request event.
     */

    let url = 'https://example.com/';
    if (headers['cloudfront-viewer-country']) {
        const countryCode = headers['cloudfront-viewer-country'][0].value;
        if (countryCode === 'TW') {
            url = 'https://tw.example.com/';
        } else if (countryCode === 'US') {
            url = 'https://us.example.com/';
```

```
        }
    }

    const response = {
        status: '302',
        statusDescription: 'Found',
        headers: {
            location: [{
                key: 'Location',
                value: url,
            }],
        },
    };
    callback(null, response);
};
```

Python

```
# This is an origin request function

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Based on the value of the CloudFront-Viewer-Country header, generate an
    HTTP status code 302 (Redirect) response, and return a country-specific
    URL in the Location header.
    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Viewer-Country header. For more information, see
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
          2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    url = 'https://example.com/'
    viewerCountry = headers.get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        if countryCode == 'TW':
            url = 'https://tw.example.com/'
        elif countryCode == 'US':
```

```
            url = 'https://us.example.com/'

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': url
            }]
        }
    }

    return response
```

示例：根据设备提供不同版本的对象

下面的示例演示如何根据用户使用的设备的类型 (例如，移动设备或平板电脑) 提供不同版本的对象。请注意以下几点：

- 您必须将您的分配配置为基于 `CloudFront-Is-*-Viewer` 标头进行缓存。有关更多信息，请参阅基于选择的请求标头进行缓存。
- CloudFront 在查看器请求事件之后添加 `CloudFront-Is-*-Viewer` 标头。要使用此示例，您必须为源请求事件创建触发器。

Node.js

```
'use strict';

/* This is an origin request function */
exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const headers = request.headers;

    /*
     * Serve different versions of an object based on the device type.
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Is-*-Viewer headers. For more information, see
     *          the following documentation:
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
```

```
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-device-type
     *       2. CloudFront adds the CloudFront-Is-*-Viewer headers after the viewer
     *          request event. To use this example, you must create a trigger for
 the
     *          origin request event.
     */

    const desktopPath = '/desktop';
    const mobilePath = '/mobile';
    const tabletPath = '/tablet';
    const smarttvPath = '/smarttv';

    if (headers['cloudfront-is-desktop-viewer']
        && headers['cloudfront-is-desktop-viewer'][0].value === 'true') {
        request.uri = desktopPath + request.uri;
    } else if (headers['cloudfront-is-mobile-viewer']
               && headers['cloudfront-is-mobile-viewer'][0].value === 'true') {
        request.uri = mobilePath + request.uri;
    } else if (headers['cloudfront-is-tablet-viewer']
               && headers['cloudfront-is-tablet-viewer'][0].value === 'true') {
        request.uri = tabletPath + request.uri;
    } else if (headers['cloudfront-is-smarttv-viewer']
               && headers['cloudfront-is-smarttv-viewer'][0].value === 'true') {
        request.uri = smarttvPath + request.uri;
    }
    console.log(`Request uri set to "${request.uri}"`);

    callback(null, request);
};
```

Python

```
# This is an origin request function
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Serve different versions of an object based on the device type.
    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Is-*-Viewer headers. For more information, see
            the following documentation:
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
```

```
            https://docs.aws.amazon.com/console/cloudfront/cache-on-device-type
          2. CloudFront adds the CloudFront-Is-*-Viewer headers after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    desktopPath = '/desktop';
    mobilePath = '/mobile';
    tabletPath = '/tablet';
    smarttvPath = '/smarttv';

    if 'cloudfront-is-desktop-viewer' in headers and headers['cloudfront-is-desktop-
viewer'][0]['value'] == 'true':
        request['uri'] = desktopPath + request['uri']
    elif 'cloudfront-is-mobile-viewer' in headers and headers['cloudfront-is-mobile-
viewer'][0]['value'] == 'true':
        request['uri'] = mobilePath + request['uri']
    elif 'cloudfront-is-tablet-viewer' in headers and headers['cloudfront-is-tablet-
viewer'][0]['value'] == 'true':
        request['uri'] = tabletPath + request['uri']
    elif 'cloudfront-is-smarttv-viewer' in headers and headers['cloudfront-is-
smarttv-viewer'][0]['value'] == 'true':
        request['uri'] = smarttvPath + request['uri']

    print("Request uri set to %s" % request['uri'])

    return request
```

基于内容的动态源选择 - 示例

本节中的示例向您展示如何可以使用 Lambda@Edge 基于请求中的信息路由到不同源。

主题

- 示例：使用源请求触发器从自定义源更改为 Amazon S3 源
- 示例：使用源请求触发器更改 Amazon S3 源区域
- 示例：使用源请求触发器从 Amazon S3 源更改为自定义源
- 示例：使用源请求触发器将流量从一个 Amazon S3 存储桶逐步转移到另一个存储桶
- 示例：使用源请求触发器根据国家/地区标头更改源域名

示例：使用源请求触发器从自定义源更改为 Amazon S3 源

此函数演示如何根据请求属性，使用源请求触发器将从中提取内容的自定义源更改为 Amazon S3 源。

Node.js

```
'use strict';

 const querystring = require('querystring');

 exports.handler = (event, context, callback) => {
     const request = event.Records[0].cf.request;

     /**
      * Reads query string to check if S3 origin should be used, and
      * if true, sets S3 origin properties.
      */

     const params = querystring.parse(request.querystring);

     if (params['useS3Origin']) {
         if (params['useS3Origin'] === 'true') {
             const s3DomainName = 'my-bucket.s3.amazonaws.com';

             /* Set S3 origin fields */
             request.origin = {
                 s3: {
                     domainName: s3DomainName,
                     region: '',
                     authMethod: 'none',
                     path: '',
                     customHeaders: {}
                 }
             };
             request.headers['host'] = [{ key: 'host', value: s3DomainName}];
         }
     }

    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    '''
    Reads query string to check if S3 origin should be used, and
    if true, sets S3 origin properties
    '''
    params = {k: v[0] for k, v in parse_qs(request['querystring']).items()}
    if params.get('useS3Origin') == 'true':
        s3DomainName = 'my-bucket.s3.amazonaws.com'

        # Set S3 origin fields
        request['origin'] = {
            's3': {
                'domainName': s3DomainName,
                'region': '',
                'authMethod': 'none',
                'path': '',
                'customHeaders': {}
            }
        }
        request['headers']['host'] = [{'key': 'host', 'value': s3DomainName}]
    return request
```

示例：使用源请求触发器更改 Amazon S3 源区域

此函数演示如何根据请求属性，使用源请求触发器更改从中提取内容的 Amazon S3 源。

在此例中，我们使用 CloudFront-Viewer-Country 标头的值将 S3 存储桶域名更新为更接近查看器的区域中的存储桶。这在多种情况下非常有用：

- 当指定的区域接近查看器所在的国家/地区时，这可以减少延迟。
- 通过确保由与发起请求所在位置的相同国家/地区的源提供数据，实现数据主权。

要使用本示例，您必须执行以下操作：

- 将您的分配配置为基于 CloudFront-Viewer-Country 标头进行缓存。有关更多信息，请参阅 基于选择的请求标头进行缓存。

- 在源请求事件中为此函数创建一个触发器。CloudFront 在查看器请求事件后添加了 `CloudFront-Viewer-Country` 标头，因此，要使用此示例，您必须确保函数对源请求执行。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    /**
     * This blueprint demonstrates how an origin-request trigger can be used to
     * change the origin from which the content is fetched, based on request
 properties.
     * In this example, we use the value of the CloudFront-Viewer-Country header
     * to update the S3 bucket domain name to a bucket in a Region that is closer to
     * the viewer.
     *
     * This can be useful in several ways:
     *      1) Reduces latencies when the Region specified is nearer to the viewer's
     *         country.
     *      2) Provides data sovereignty by making sure that data is served from an
     *         origin that's in the same country that the request came from.
     *
     * NOTE: 1. You must configure your distribution to cache based on the
     *          CloudFront-Viewer-Country header. For more information, see
     *          https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-
headers
     *       2. CloudFront adds the CloudFront-Viewer-Country header after the
 viewer
     *          request event. To use this example, you must create a trigger for
 the
     *          origin request event.
     */

    const countryToRegion = {
        'DE': 'eu-central-1',
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    };
```

```
    if (request.headers['cloudfront-viewer-country']) {
        const countryCode = request.headers['cloudfront-viewer-country'][0].value;
        const region = countryToRegion[countryCode];

        /**
         * If the viewer's country is not in the list you specify, the request
         * goes to the default S3 bucket you've configured.
         */
        if (region) {
            /**
             * If you've set up OAI, the bucket policy in the destination bucket
             * should allow the OAI GetObject operation, as configured by default
             * for an S3 origin with OAI. Another requirement with OAI is to provide
             * the Region so it can be used for the SIGV4 signature. Otherwise, the
             * Region is not required.
             */
            request.origin.s3.region = region;
            const domainName = `my-bucket-in-${region}.s3.amazonaws.com`;
            request.origin.s3.domainName = domainName;
            request.headers['host'] = [{ key: 'host', value: domainName }];
        }
    }

    callback(null, request);
};
```

Python

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    '''
    This blueprint demonstrates how an origin-request trigger can be used to
    change the origin from which the content is fetched, based on request
 properties.
    In this example, we use the value of the CloudFront-Viewer-Country header
    to update the S3 bucket domain name to a bucket in a Region that is closer to
    the viewer.

    This can be useful in several ways:
        1) Reduces latencies when the Region specified is nearer to the viewer's
            country.
```

```
        2) Provides data sovereignty by making sure that data is served from an
            origin that's in the same country that the request came from.

    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Viewer-Country header. For more information, see
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
          2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    countryToRegion = {
        'DE': 'eu-central-1',
        'IE': 'eu-west-1',
        'GB': 'eu-west-2',
        'FR': 'eu-west-3',
        'JP': 'ap-northeast-1',
        'IN': 'ap-south-1'
    }

    viewerCountry = request['headers'].get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        region = countryToRegion.get(countryCode)

        # If the viewer's country in not in the list you specify, the request
        # goes to the default S3 bucket you've configured
        if region:
            '''
            If you've set up OAI, the bucket policy in the destination bucket
            should allow the OAI GetObject operation, as configured by default
            for an S3 origin with OAI. Another requirement with OAI is to provide
            the Region so it can be used for the SIGV4 signature. Otherwise, the
            Region is not required.
            '''
            request['origin']['s3']['region'] = region
            domainName = 'my-bucket-in-%s.s3.amazonaws.com' % region
            request['origin']['s3']['domainName'] = domainName
            request['headers']['host'] = [{'key': 'host', 'value': domainName}]

    return request
```

示例：使用源请求触发器从 Amazon S3 源更改为自定义源

该函数演示如何根据请求属性，使用源请求触发器更改从中提取内容的自定义源。

Node.js

```
'use strict';

const querystring = require('querystring');

 exports.handler = (event, context, callback) => {
     const request = event.Records[0].cf.request;

     /**
      * Reads query string to check if custom origin should be used, and
      * if true, sets custom origin properties.
      */

     const params = querystring.parse(request.querystring);

     if (params['useCustomOrigin']) {
         if (params['useCustomOrigin'] === 'true') {

             /* Set custom origin fields*/
             request.origin = {
                 custom: {
                     domainName: 'www.example.com',
                     port: 443,
                     protocol: 'https',
                     path: '',
                     sslProtocols: ['TLSv1', 'TLSv1.1'],
                     readTimeout: 5,
                     keepaliveTimeout: 5,
                     customHeaders: {}
                 }
             };
             request.headers['host'] = [{ key: 'host', value: 'www.example.com'}];
         }
     }
    callback(null, request);
};
```

Python

```
from urllib.parse import parse_qs

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    # Reads query string to check if custom origin should be used, and
    # if true, sets custom origin properties

    params = {k: v[0] for k, v in parse_qs(request['querystring']).items()}

    if params.get('useCustomOrigin') == 'true':
            # Set custom origin fields
            request['origin'] = {
                'custom': {
                    'domainName': 'www.example.com',
                    'port': 443,
                    'protocol': 'https',
                    'path': '',
                    'sslProtocols': ['TLSv1', 'TLSv1.1'],
                    'readTimeout': 5,
                    'keepaliveTimeout': 5,
                    'customHeaders': {}
                }
            }
            request['headers']['host'] = [{'key': 'host', 'value':
 'www.example.com'}]

    return request
```

示例：使用源请求触发器将流量从一个 Amazon S3 存储桶逐步转移到另一个存储桶

此函数演示如何以可控的方式将流量从一个 Amazon S3 存储桶逐步转移到另一个存储桶。

Node.js

```
'use strict';

    function getRandomInt(min, max) {
        /* Random number is inclusive of min and max*/
        return Math.floor(Math.random() * (max - min + 1)) + min;
```

```
 }

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;
    const BLUE_TRAFFIC_PERCENTAGE = 80;

    /**
      * This Lambda function demonstrates how to gradually transfer traffic from
      * one S3 bucket to another in a controlled way.
      * We define a variable BLUE_TRAFFIC_PERCENTAGE which can take values from
      * 1 to 100. If the generated randomNumber less than or equal to
 BLUE_TRAFFIC_PERCENTAGE, traffic
      * is re-directed to blue-bucket. If not, the default bucket that we've
 configured
      * is used.
      */

    const randomNumber = getRandomInt(1, 100);

if (randomNumber <= BLUE_TRAFFIC_PERCENTAGE) {
         const domainName = 'blue-bucket.s3.amazonaws.com';
         request.origin.s3.domainName = domainName;
         request.headers['host'] = [{ key: 'host', value: domainName}];
     }
    callback(null, request);
};
```

Python

```
import math
import random

def getRandomInt(min, max):
    # Random number is inclusive of min and max
    return math.floor(random.random() * (max - min + 1)) + min

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    BLUE_TRAFFIC_PERCENTAGE = 80

    '''
    This Lambda function demonstrates how to gradually transfer traffic from
    one S3 bucket to another in a controlled way.
```

```
    We define a variable BLUE_TRAFFIC_PERCENTAGE which can take values from
    1 to 100. If the generated randomNumber less than or equal to
 BLUE_TRAFFIC_PERCENTAGE, traffic
    is re-directed to blue-bucket. If not, the default bucket that we've configured
    is used.
    '''

    randomNumber = getRandomInt(1, 100)

    if randomNumber <= BLUE_TRAFFIC_PERCENTAGE:
        domainName = 'blue-bucket.s3.amazonaws.com'
        request['origin']['s3']['domainName'] = domainName
        request['headers']['host'] = [{'key': 'host', 'value': domainName}]

    return request
```

示例：使用源请求触发器根据国家/地区标头更改源域名

此函数演示了如何根据 CloudFront-Viewer-Country 标头更改源域名，这样可以从接近查看器所在的国家/地区的源提供内容。

为您的分配实施此功能可能有类似于下面的好处：

- 在指定的区域接近查看器所在的国家/地区时减少延迟。
- 确保由请求发起位置所在的同一国家/地区内的源提供数据，从而实现数据主权。

请注意，要启用此功能，您必须配置分配以根据 CloudFront-Viewer-Country 标头进行缓存。有关更多信息，请参阅 the section called “基于选择的请求标头进行缓存”。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
     const request = event.Records[0].cf.request;

  if (request.headers['cloudfront-viewer-country']) {
        const countryCode = request.headers['cloudfront-viewer-country'][0].value;
        if (countryCode === 'GB' || countryCode === 'DE' || countryCode === 'IE' )
 {
            const domainName = 'eu.example.com';
```

```
            request.origin.custom.domainName = domainName;
            request.headers['host'] = [{key: 'host', value: domainName}];
        }
    }

    callback(null, request);
};
```

Python

```
def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']

    viewerCountry = request['headers'].get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        if countryCode == 'GB' or countryCode == 'DE' or countryCode == 'IE':
            domainName = 'eu.example.com'
            request['origin']['custom']['domainName'] = domainName
            request['headers']['host'] = [{'key': 'host', 'value': domainName}]
    return request
```

更新错误状态 - 示例

本节中的示例提供有关如何可以使用 Lambda@Edge 更改返回给用户的错误状态的指导。

主题

- 示例：使用源响应触发器将错误状态代码更新为 200
- 示例：使用源响应触发器将错误状态代码更新为 302

示例：使用源响应触发器将错误状态代码更新为 200

该函数演示了在下列情况中，如何将响应状态更新为 200 并生成静态正文内容以返回到查看器：

- 函数在源响应中触发。
- 来自源服务器的响应状态是错误状态代码（4xx 或 5xx）。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const response = event.Records[0].cf.response;

    /**
     * This function updates the response status to 200 and generates static
     * body content to return to the viewer in the following scenario:
     * 1. The function is triggered in an origin response
     * 2. The response status from the origin server is an error status code (4xx or
 5xx)
     */

    if (response.status >= 400 && response.status <= 599) {
        response.status = 200;
        response.statusDescription = 'OK';
        response.body = 'Body generation example';
    }

    callback(null, response);
};
```

Python

```
def lambda_handler(event, context):
    response = event['Records'][0]['cf']['response']

    '''
    This function updates the response status to 200 and generates static
    body content to return to the viewer in the following scenario:
    1. The function is triggered in an origin response
    2. The response status from the origin server is an error status code (4xx or
 5xx)
    '''

    if int(response['status']) >= 400 and int(response['status']) <= 599:
        response['status'] = 200
        response['statusDescription'] = 'OK'
        response['body'] = 'Body generation example'
    return response
```

示例：使用源响应触发器将错误状态代码更新为 302

该函数演示了如何将 HTTP 状态代码更新为 302，以重定向到配置了不同源的其他路径（缓存行为）。请注意以下几点：

- 函数在源响应中触发。
- 来自源服务器的响应状态是错误状态代码（4xx 或 5xx）。

Node.js

```
'use strict';

exports.handler = (event, context, callback) => {
    const response = event.Records[0].cf.response;
    const request = event.Records[0].cf.request;

    /**
     * This function updates the HTTP status code in the response to 302, to
 redirect to another
     * path (cache behavior) that has a different origin configured. Note the
 following:
     * 1. The function is triggered in an origin response
     * 2. The response status from the origin server is an error status code (4xx or
 5xx)
     */

    if (response.status >= 400 && response.status <= 599) {
        const redirect_path = `/plan-b/path?${request.querystring}`;

        response.status = 302;
        response.statusDescription = 'Found';

        /* Drop the body, as it is not required for redirects */
        response.body = '';
        response.headers['location'] = [{ key: 'Location', value: redirect_path }];
    }

    callback(null, response);
};
```

Python

```
def lambda_handler(event, context):
    response = event['Records'][0]['cf']['response']
    request = event['Records'][0]['cf']['request']

    '''
    This function updates the HTTP status code in the response to 302, to redirect
 to another
    path (cache behavior) that has a different origin configured. Note the
 following:
    1. The function is triggered in an origin response
    2. The response status from the origin server is an error status code (4xx or
 5xx)
    '''

    if int(response['status']) >= 400 and int(response['status']) <= 599:
        redirect_path = '/plan-b/path?%s' % request['querystring']

        response['status'] = 302
        response['statusDescription'] = 'Found'

        # Drop the body as it is not required for redirects
        response['body'] = ''
        response['headers']['location'] = [{'key': 'Location', 'value':
 redirect_path}]

    return response
```

访问请求正文 - 示例

本节中的示例说明如何使用 Lambda@Edge 来处理 POST 请求。

> **Note**
>
> 要使用这些示例，必须在分配的 Lambda 函数关联中启用 include body（包含正文）选项。默认情况下，将不会启用此选项。
>
> - 要在 CloudFront 控制台中启用此设置，请选中 Lambda 函数关联中的包含正文复选框。

- 要在 CloudFront API 中或使用 AWS CloudFormation 启用此设置，请在 `LambdaFunctionAssociation` 中将 `IncludeBody` 字段设置为 `true`。

主题

示例：使用请求触发器读取 HTML 表单

该函数说明了如何处理 HTML 表单（Web 表单）生成的 POST 请求的正文，例如“联系我们”表单。例如，您可能具有如下所示的 HTML 表单：

```
<html>
  <form action="https://example.com" method="post">
    Param 1: <input type="text" name="name1"><br>
    Param 2: <input type="text" name="name2"><br>
    input type="submit" value="Submit">
  </form>
</html>
```

对于后面的函数示例，必须在 CloudFront 查看器请求或源请求中触发该函数。

Node.js

```
'use strict';

const querystring = require('querystring');

/**
 * This function demonstrates how you can read the body of a POST request
 * generated by an HTML form (web form). The function is triggered in a
 * CloudFront viewer request or origin request event type.
 */

exports.handler = (event, context, callback) => {
    const request = event.Records[0].cf.request;

    if (request.method === 'POST') {
        /* HTTP body is always passed as base64-encoded string. Decode it. */
```

```
        const body = Buffer.from(request.body.data, 'base64').toString();

        /* HTML forms send the data in query string format. Parse it. */
        const params = querystring.parse(body);

        /* For demonstration purposes, we only log the form fields here.
         * You can put your custom logic here. For example, you can store the
         * fields in a database, such as Amazon DynamoDB, and generate a response
         * right from your Lambda@Edge function.
         */
        for (let param in params) {
            console.log(`For "${param}" user submitted "${params[param]}".\n`);
        }
    }
    return callback(null, request);
};
```

Python

```
import base64
from urllib.parse import parse_qs

'''
Say there is a POST request body generated by an HTML such as:

<html>
<form action="https://example.com" method="post">
    Param 1: <input type="text" name="name1"><br>
    Param 2: <input type="text" name="name2"><br>
    input type="submit" value="Submit">
</form>
</html>

'''

'''
This function demonstrates how you can read the body of a POST request
generated by an HTML form (web form). The function is triggered in a
CloudFront viewer request or origin request event type.
'''

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
```

```
    if request['method'] == 'POST':
        # HTTP body is always passed as base64-encoded string. Decode it
        body = base64.b64decode(request['body']['data'])

        # HTML forms send the data in query string format. Parse it
        params = {k: v[0] for k, v in parse_qs(body).items()}

        '''
        For demonstration purposes, we only log the form fields here.
        You can put your custom logic here. For example, you can store the
        fields in a database, such as Amazon DynamoDB, and generate a response
        right from your Lambda@Edge function.
        '''
        for key, value in params.items():
            print("For %s use submitted %s" % (key, value))

    return request
```

示例：使用请求触发器修改 HTML 表单

该函数说明了如何修改 HTML 表单（Web 表单）生成的 POST 请求的正文。在 CloudFront 查看器请求或源请求中触发该函数。

Node.js

```
'use strict';

const querystring = require('querystring');

exports.handler = (event, context, callback) => {
    var request = event.Records[0].cf.request;
    if (request.method === 'POST') {
        /* Request body is being replaced. To do this, update the following
        /* three fields:
         *    1) body.action to 'replace'
         *    2) body.encoding to the encoding of the new data.
         *
         *       Set to one of the following values:
         *
         *           text - denotes that the generated body is in text format.
         *               Lambda@Edge will propagate this as is.
```

```
         *           base64 - denotes that the generated body is base64 encoded.
         *               Lambda@Edge will base64 decode the data before sending
         *               it to the origin.
         *    3) body.data to the new body.
         */
        request.body.action = 'replace';
        request.body.encoding = 'text';
        request.body.data = getUpdatedBody(request);
    }
    callback(null, request);
};

function getUpdatedBody(request) {
    /* HTTP body is always passed as base64-encoded string. Decode it. */
    const body = Buffer.from(request.body.data, 'base64').toString();

    /* HTML forms send data in query string format. Parse it. */
    const params = querystring.parse(body);

    /* For demonstration purposes, we're adding one more param.
     *
     * You can put your custom logic here. For example, you can truncate long
     * bodies from malicious requests.
     */
    params['new-param-name'] = 'new-param-value';
    return querystring.stringify(params);
}
```

Python

```
import base64
from urllib.parse import parse_qs, urlencode

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    if request['method'] == 'POST':
        '''
        Request body is being replaced. To do this, update the following
        three fields:
            1) body.action to 'replace'
            2) body.encoding to the encoding of the new data.

            Set to one of the following values:
```

```
                text - denotes that the generated body is in text format.
                    Lambda@Edge will propagate this as is.
                base64 - denotes that the generated body is base64 encoded.
                    Lambda@Edge will base64 decode the data before sending
                    it to the origin.
            3) body.data to the new body.
        '''
        request['body']['action'] = 'replace'
        request['body']['encoding'] = 'text'
        request['body']['data'] = getUpdatedBody(request)
    return request

def getUpdatedBody(request):
    # HTTP body is always passed as base64-encoded string. Decode it
    body = base64.b64decode(request['body']['data'])

    # HTML forms send data in query string format. Parse it
    params = {k: v[0] for k, v in parse_qs(body).items()}

    # For demonstration purposes, we're adding one more param

    # You can put your custom logic here. For example, you can truncate long
    # bodies from malicious requests
    params['new-param-name'] = 'new-param-value'
    return urlencode(params)
```

边缘函数的限制

以下主题描述了适用于 CloudFront Functions 和 Lambda@Edge 的限制。一些限制适用于所有边缘函数，而另一些限制仅适用于 CloudFront Functions 或 Lambda@Edge。

有关配额（以前被称为限制）的更多信息，请参阅 CloudFront Functions 的配额 和 有关 Lambda@Edge 的配额。

主题

- 所有边缘函数的限制
- 对 CloudFront Functions 的限制
- 对 Lambda@Edge 的限制

所有边缘函数的限制

以下限制适用于所有边缘函数，包括 CloudFront Functions 和 Lambda@Edge。

主题

AWS 账户所有权

要将边缘函数与 CloudFront 分配相关联，该函数和分配必须属于相同的 AWS 账户。

组合 CloudFront Functions 与 Lambda@Edge

对于给定缓存行为，将适用以下限制将适用：

- 每个事件类型（查看器请求、源请求、源响应和查看器响应）只能有一个边缘函数关联。
- 您不能在查看器事件（查看器请求和查看器响应）中组合 CloudFront Functions 和 Lambda@Edge。

允许使用边缘函数的所有其他组合。下表介绍了允许的组合。

		CloudFront Functions	
		查看器请求	查看器响应
Lambda@Edge	查看器请求	不允许	不允许
	源请求	已允许	已允许

	源响应	已允许	已允许
	查看器响应	不允许	不允许

HTTP 状态代码

如果源返回 400 或更高的 HTTP 状态代码，则 CloudFront 不会对查看器响应事件调用边缘函数。

用于源响应事件的 Lambda@Edge 函数被调用于全部源响应，包括源返回 400 或更高的 HTTP 状态代码。有关更多信息，请参阅 更新源响应触发器中的 HTTP 响应。

HTTP 标头

不允许使用某些 HTTP 标头，这意味着这些标头不会向边缘函数公开，并且函数无法添加它们。其他标头是只读的，这意味着函数可以读取它们，但不能添加或修改它们。

主题

- 不允许使用的标头
- 只读标头

不允许使用的标头

以下 HTTP 标头不会向边缘函数公开，并且函数无法添加它们。如果您的函数添加这些标头的其中之一，则请求将无法通过 CloudFront 验证，并且 CloudFront 会将 HTTP 状态代码 502（无效网关）返回给查看器。

- `Connection`
- `Expect`
- `Keep-Alive`
- `Proxy-Authenticate`
- `Proxy-Authorization`
- `Proxy-Connection`
- `Trailer`
- `Upgrade`
- `X-Accel-Buffering`

- X-Accel-Charset
- X-Accel-Limit-Rate
- X-Accel-Redirect
- X-Amz-Cf-*
- X-Amzn-Auth
- X-Amzn-Cf-Billing
- X-Amzn-Cf-Id
- X-Amzn-Cf-Xff
- X-Amzn-Errortype
- X-Amzn-Fle-Profile
- X-Amzn-Header-Count
- X-Amzn-Header-Order
- X-Amzn-Lambda-Integration-Tag
- X-Amzn-RequestId
- X-Cache
- X-Edge-*
- X-Forwarded-Proto
- X-Real-IP

只读标头

以下标头是只读的。您的函数可以读取这些标头，并将其用作函数逻辑输入，但无法更改这些值。如果您的函数添加或编辑一个只读标头，请求将无法通过 CloudFront 验证，并且 CloudFront 将 HTTP 状态代码 502（无效网关）返回到查看器。

查看器请求事件中的只读标头

以下标头在查看器请求事件中为只读标头。

- Content-Length
- Host
- Transfer-Encoding
- Via

源请求事件中的只读标头（仅限 Lambda@Edge）

以下标头在源请求事件中是只读的，它们仅存在于 Lambda@Edge 中。

- Accept-Encoding
- Content-Length
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Transfer-Encoding
- Via

源响应事件中的只读标头（仅限 Lambda@Edge）

以下标头在源响应事件中是只读的，它们仅存在于 Lambda@Edge 中。

- Transfer-Encoding
- Via

查看器响应事件中的只读标头

以下标头在查看器响应事件中为只读标头（对于 CloudFront Functions 和 Lambda@Edge）

- Warning
- Via

以下标头在查看器响应事件中为只读标头（对于 Lambda@Edge）。

- Content-Length
- Content-Encoding
- Transfer-Encoding

查询字符串

以下限制适用于在请求 URI 中读取、更新或创建查询字符串的函数。

- （仅限 Lambda@Edge）要访问源请求或源响应函数中的查询字符串，您的缓存策略或源请求策略必须针对查询字符串设置为 All。
- 函数可以为查看器请求和源请求事件创建或更新查询字符串（源请求事件仅在 Lambda@Edge 中存在）。
- 函数可以为源响应和查看器响应事件读取查询字符串，但不能创建或更新查询字符串（源响应事件仅在 Lambda@Edge 中存在）。
- 如果函数创建或更新查询字符串，则具有以下限制：
  - 查询字符串不能包含空格、控制字符或片段标识符 (#)。
  - URI（包括查询字符串）的总大小必须小于 8192 个字符。
  - 建议您对 URI 和查询字符串使用百分号编码。有关更多信息，请参阅 URI 和查询字符串编码。

URI

如果某个函数更改请求的 URI，则这样不会更改该请求或该请求转发到的源的缓存行为。

URI（包括查询字符串）的总大小必须小于 8192 个字符。

URI 和查询字符串编码

传递给边缘函数的 URI 和查询字符串值是 UTF-8 编码。您的函数应该对其返回的 URI 和查询字符串值使用 UTF-8 编码。百分号编码与 UTF-8 编码兼容。

下面的列表说明了 CloudFront 如何处理 URI 和查询字符串值编码：

- 如果请求中的值是 UTF-8 编码，则 CloudFront 会将值转发给您的函数，而不会更改它们。
- 如果请求中的值为 ISO-8859-1 编码，CloudFront 会将值转换为 UTF-8 编码，然后再将值转发给您的函数。
- 如果请求中的值使用任何其他字符编码方式进行编码，则 CloudFront 会假定它们是 ISO-8859-1 编码，并尝试从 ISO-8859-1 编码转换为 UTF-8 编码。

> **Important**
>
> 转换后字符可能是原始请求中的值的不准确解释。这可能会导致您的函数或源生成意外结果。

CloudFront 转发到源的 URI 和查询字符串值取决于函数是否更改了值：

- 如果函数没有更改 URI 或查询字符串，CloudFront 将其在请求中收到的值转发到源。
- 如果函数更改了 URI 或查询字符串，CloudFront 转发 UTF-8 编码的值。

Microsoft Smooth Streaming

您无法将用于流式传输媒体文件（已转码为 Microsoft Smooth Streaming 格式）的 CloudFront 分配与边缘函数一同使用。

标记

您无法将标签添加到边缘函数中。要了解有关在 CloudFront 中进行标记的更多信息，请参阅标记分配。

对 CloudFront Functions 的限制

以下限制仅适用于 CloudFront Functions。

有关配额（以前称为限制）的更多信息，请参阅CloudFront Functions 的配额。

日志

CloudFront Functions 中的函数日志被截断为 10 KB。

请求正文

CloudFront Functions 无法访问 HTTP 请求的正文。

使用 CloudFront KeyValueStore API 时的区域 AWS Security Token Service 端点

当您使用带临时安全凭证的签名版本 4A（SigV4A）调用 CloudFront KeyValueStore API 时（例如，在使用 AWS Identity and Access Management（IAM）角色时），请确保从 AWS STS 中的区域端点请求临时凭证。如果您对 AWS STS（`sts.amazonaws.com`）使用全局端点，AWS STS 将从全局端点生成 SigV4A 不支持的临时凭证。因此，您将会收到身份验证错误。要解决此问题，请使用《IAM 用户指南》中列出的任何适用于 AWS STS 的区域端点。如果您将 SAML 配置为使用 AWS STS 区域端点，请参阅如何使用区域 SAML 端点进行故障转移博客文章。

运行时

CloudFront Functions 运行时环境不支持动态代码评估，它限制了对网络、文件系统和计时器的访问。有关更多信息，请参阅 受限功能。

> **Note**
>
> 要使用 CloudFront keyValueStore，您的 CloudFront 函数必须使用 JavaScript 运行时 2.0。

计算利用率

CloudFront Functions 对运行时间有限制，测量方式为计算利用率。计算利用率是介于 0 到 100 之间的数字，表示函数运行所花费的时间占最大允许时间的百分比。例如，计算利用率为 35 表示函数在最大允许时间的 35% 内完成。

当您测试函数时，您可以在测试事件的输出中看到计算利用率值。对于生产函数，您可以在 CloudFront 控制台中的监控页面或在 CloudWatch 中查看计算利用率指标。

对 Lambda@Edge 的限制

以下限制仅适用于 Lambda@Edge。

有关配额的信息，请参阅有关 Lambda@Edge 的配额。

DNS 解析

CloudFront 会先对源域名执行 DNS 解析，然后 再执行源请求 Lambda@Edge 函数。如果您的域的 DNS 服务出现问题，并且 CloudFront 无法解析域名以获取 IP 地址，将不会调用您的 Lambda@Edge 函数。CloudFront 会将 HTTP 状态代码 502（无效网关）返回到客户端。有关更多信息，请参阅 DNS 错误（`NonS3OriginDnsError`）。

有关管理 DNS 故障转移的更多信息，请参阅《Amazon Route 53 开发人员指南》中的配置 DNS 故障转移。

HTTP 状态代码

查看器响应事件的 Lambda@Edge 函数无法修改响应的 HTTP 状态代码，无论响应是来自源还是 CloudFront 缓存。

Lambda 函数版本

您必须使用带编号的 Lambda 函数，而不是 `$LATEST` 或别名。

Lambda 区域

Lambda 函数必须位于美国东部（弗吉尼亚州北部）区域。

Lambda 角色权限

与 Lambda 函数关联的 IAM 执行角色必须由服务委托人 `lambda.amazonaws.com` 和 `edgelambda.amazonaws.com` 担任。有关更多信息，请参阅 设置 Lambda@Edge 的 IAM 权限和角色。

Lambda 功能

Lambda@Edge 不支持以下 Lambda 功能：

- 自动（默认设置）以外的 Lambda 运行时管理配置。
- 将您的 Lambda 函数配置为访问 VPC 内的资源
- Lambda 函数死信队列。
- Lambda 环境变量（自动支持的预留环境变量除外）
- 具有 AWS Lambda 层的 Lambda 函数
- 使用 AWS X-Ray
- Lambda 预配置并发

> **Note**
>
> Lambda@Edge 函数具有与 Lambda 函数相同的区域并发功能。但是，当增加并发 Lambda@Edge 执行的配额时，所有复制 Lambda@Edge 函数的 AWS 区域位置的配额都会增加。有关更多信息，请参阅 有关 Lambda@Edge 的配额。

- 定义为容器映像的 Lambda 函数
- 使用 arm64 架构的 Lambda 函数
- 短暂存储超过 512 MB 的 Lambda 函数。
- 以 JSON 结构化格式捕获 Lambda 函数日志
- 控制 Lambda 函数日志的日志级别粒度
- 设置 Lambda 将日志发送到哪个 Amazon CloudWatch 日志组

支持的运行时

Lambda@Edge 支持具有以下运行时的 Lambda 函数：

Node.js	Python
• Node.js 20 • Node.js 18 • Node.js 16[1] • Node.js 14[2] • Node.js 12[2] • Node.js 10[2] • Node.js 8[2] • Node.js 6[2]	• Python 3.12 • Python 3.11 • Python 3.10 • Python 3.9 • Python 3.8 • Python 3.7

[1]此版本的 Node.js 使用寿命已结束，AWS Lambda 很快会将其弃用。

[2]此版本的 Node.js 使用寿命已结束，AWS Lambda 已将其完全弃用。

您无法使用已弃用的 Node.js 版本创建或更新函数。您只能将现有函数与这些版本及 CloudFront 分配相关联。使用与分配关联的这些版本的函数将继续运行。但是，建议将您的函数转移到更新版本的 Node.js。有关更多信息，请参阅《AWS Lambda 开发人员指南》中的运行时弃用策略和 GitHub 上的 Node.js 发布计划。

> **Tip**
>
> 作为最佳实践，请使用所提供的运行时的最新版本来改进性能和新增功能。

CloudFront 标头

Lambda@Edge 函数可以读取、编辑、删除或添加 添加 CloudFront 请求标头 中列出的任何 CloudFront 标头。

> **注意**
>
> - 如果您希望 CloudFront 添加这些标头，则必须将 CloudFront 配置为使用缓存策略或源请求策略来添加它们。
> - CloudFront 在查看器请求事件之后 添加标题，这意味着这些标头在查看器请求函数中不可用于 Lambda@Edge 函数。这些标头仅适用于源请求和源响应中的 Lambda@Edge 函数。

- 如果查看器请求包含具有这些名称的标头，并且您将 CloudFront 配置为使用缓存策略或者源请求策略添加这些标头，则 CloudFront 会覆盖查看器请求中的标头值。面向查看器的函数可以看到查看器请求中的标头值，而面向源的函数则看到 CloudFront 添加的标头值。
- 如果查看器请求函数添加 `CloudFront-Viewer-Country` 标头，它将无法通过验证，并且 CloudFront 会将 HTTP 状态代码 502（无效网关）返回到查看器。

具有 Include Body（包含正文）选项的请求正文的限制

在选择包含正文选项以向您的 Lambda@Edge 函数公开请求正文时，以下信息和大小配额适用于公开或替换的正文部分。

- CloudFront 总是对请求正文进行 base64 编码，然后再将其公开给 Lambda@Edge。
- 如果请求正文很大，则 CloudFront 在将正文公开给 Lambda@Edge 之前将其截断，如下所示：
  - 对于查看器请求事件，正文将截断为 40 KB。
  - 对于源请求事件，正文将截断为 1 MB。
- 如果以只读方式访问请求正文，则 CloudFront 将完整的原始请求正文发送到源。
- 如果您的 Lambda@Edge 函数替换请求正文，则以下大小配额适用于该函数返回的正文：
  - 如果 Lambda@Edge 函数以纯文本形式返回正文：
    - 对于查看器请求事件，正文将截断为 40 KB。
    - 对于源请求事件，正文将截断为 1 MB。
  - 如果 Lambda@Edge 函数以 base64 编码文本形式返回正文：
    - 对于查看器请求事件，正文将截断为 53.2 KB。
    - 对于源请求事件，正文将截断为 1.33 MB。

响应超时和保持连接超时（仅自定义源）

如果您使用 Lambda@Edge 函数为分配源设置响应超时或保持连接超时，请确认您指定了您的源可以支持的值。有关更多信息，请参阅 响应和保持连接超时限额。

报告、指标和日志

CloudFront 提供了多种选项来报告、监控和记录您的 CloudFront 资源：

- 您可以查看和下载报告，以了解您的 CloudFront 分配的使用情况和活动，包括计费报告、缓存统计、热门内容和主要引用站点。
- 您可以直接在 CloudFront 控制台中或使用 Amazon CloudWatch 监控和跟踪 CloudFront，包括您的边缘计算函数。CloudFront 向 CloudWatch 发送分配和边缘函数（包括 Lambda@Edge 和 CloudFront Functions）的各种指标。
- 您可以使用标准日志或实时日志查看 CloudFront 分配收到的查看器请求的日志。除了查看器请求日志外，您还可以使用 CloudWatch Logs 获取边缘函数（包括 Lambda@Edge 和 CloudFront Functions）的日志。您还可以使用 AWS CloudTrail 获取您的 AWS 账户中 CloudFront API 活动的日志。
- 可以使用 AWS Config 跟踪 CloudFront 资源的配置更改。

有关每个选项的更多信息，请参阅以下主题。

主题

CloudFront 的 AWS 账单和使用率报告

AWS 为 CloudFront 提供了两份使用率报告：

- AWS账单报告从宏观层面显示了您所用的AWS 服务的所有活动（包括 CloudFront）。
- AWS使用情况报告概括了特定服务按小时、天或月汇总的活动情况。它还包括提供了 CloudFront 使用情况图示的使用情况图表。

> **Note**
>
> 与其他AWS 服务一样，CloudFront 只对您使用的部分收费。有关更多信息，请参阅 CloudFront 定价。

主题

查看 CloudFront 的AWS账单报告

您可在AWS Billing and Cost Management控制台的账单页面上，查看您的AWS使用情况和费用摘要。

查看AWS账单报告

1. 访问 https://console.aws.amazon.com/billing/，登录 AWS Management Console 并打开 AWS Billing 控制台。
2. 在导航窗格上，选择账单。
3. 选择一个账单周期（例如，2023 年 8 月）。
4. 在按服务计费选项卡上，选择 CloudFront，然后展开全局或AWS 区域名称。
5. 要以 CSV 格式下载详细账单报告，请选择全部下载为 CSV。

有关AWS账单的更多信息，请参阅《AWS Billing 用户指南》中的查看您的账单。

账单报告包含适用于 CloudFront 的以下值：

- ProductCode – `AmazonCloudFront`
- UsageType – 以下值之一：
  - 标识数据传输类型的代码
  - `Invalidations`
  - `Executions-CloudFrontFunctions`
  - `KeyValueStore-APIOperations`
  - `KeyValueStore-EdgeReads`

  - `RealTimeLog-KinesisDataStream`
  - `SSL-Cert-Custom`
- ItemDescription – UsageType 的账单费率的说明。
- UsageStart Date 和 UsageEndDate – 使用世界协调时（UTC，Coordinated Universal Time）表示的使用日期。
- UsageQuantity – 以下值之一：
  - 指定时间段内的请求数
  - 传输的数据量（以 GB 为单位）
  - 无效的对象的数量
  - 您让 SSL 证书与已启用 CloudFront 分配关联的分摊月份的总数。例如，如果您让一个证书与一个已启用分配关联一个月，并让另一个证书与另一个已启用分配关联半个月，该值将为 1.5。

查看 CloudFront 的AWS使用情况报告

AWS提供了详细程度高于账单报告但低于 CloudFront 访问日志记录的 CloudFront 使用情况报告。该使用情况报告提供了按小时、天或月汇总的使用数据；并按区域和使用类型（例如传出澳大利亚区域的数据）列出了操作。

查看AWS使用情况报告

1. 访问 https://console.aws.amazon.com/billing/，登录 AWS Management Console 并打开 AWS Billing 控制台。
2. 在导航窗格中，选择成本和使用情况报告。
3. 在 AWS 使用情况报告部分下面，选择创建使用情况报告。
4. 在下载使用报告页面的服务下，选择 Amazon CloudFront
5. 选择使用类型。
6. 选择操作。
7. 选择报告的时间段。如果选择自定义日期范围，您需要手动指定报告的日期范围。
8. 在报告精细程度下面，选择每小时、每天或每月。
9. 选择下载，然后选择 XML 报告或 CSV 报告。

有关AWS使用情况报告的更多信息，请参阅https://docs.aws.amazon.com/cur/latest/userguide/usage-report.html《AWS数据导出用户指南》中的AWS Data Exports使用情况报告。

CloudFront 使用情况报告包含以下值：

- Service (服务) – AmazonCloudFront
- Operation – HTTP 方法。值包括 DELETE、GET、HEAD、OPTIONS、PATCH、POST 和 PUT。
- UsageType – 以下值之一：
  - 标识数据传输类型的代码
  - Invalidations
  - Executions-CloudFrontFunctions
  - KeyValueStore-APIOperations
  - KeyValueStore-EdgeReads
  - RealTimeLog-KinesisDataStream
  - SSL-Cert-Custom
- Resource – 与使用情况关联的 CloudFront 分配的 ID，或您关联到 CloudFront 分配的 SSL 证书的证书 ID。
- StartTime/EndTime – 使用协调世界时（UTC）表示的使用日期。
- UsageValue – 1) 在指定的时间段内的请求数，或者 2) 传输的数据量（以字节为单位）。

如果您将 Amazon S3 用作 CloudFront 的源，还请考虑运行 Amazon S3 的使用情况报告。但是，如果您将 Amazon S3 用于作为您的 CloudFront 分配的源的其他目的，则可能无法确定哪个部分适用于您的 CloudFront 使用。

> **Tip**
>
> 对于 CloudFront 收到的针对您对象的每一个请求，如需获得详细信息，请打开您的分配的 CloudFront 访问日志。有关更多信息，请参阅 the section called “使用标准日志（访问日志）”。

有关了解报告中的 CloudFront 费用和使用类型的更多信息，请参阅the section called “解释 CloudFront 的AWS账单和使用情况报告”。

解释 CloudFront 的AWS账单和使用情况报告

在获得账单报告和使用情况报告后，您可以通过本主题来了解如何解释账单上显示的每项 CloudFront 费用，以及每项费用对应的使用类型。本主题介绍了两个报告中可能出现的代码和AWS 区域缩写。

两列中的大多数代码都包含一个由两个字母组成的缩写，该缩写指示了活动的发生位置。在下表中，代码中的 *region* 在您的 AWS 账单和使用情况报告中将被下面其中一个由两个字母组成的缩写取代：

- AP：香港、菲律宾、韩国、 台湾地区和新加坡（亚太地区）
- AU：澳大利亚
- CA：加拿大
- EU：欧洲和以色列
- IN：印度
- JP：日本
- ME：中东
- SA：南美洲
- US：美国
- ZA：南非

有关各AWS 区域定价的更多信息，请参阅 Amazon CloudFront 定价。

> **注意**
>
> - 该表未包含将您的对象从 Amazon S3 存储桶传输到 CloudFront 边缘站点所产生的费用。这些费用（如果有）显示在您的 AWS 账单中的 AWS 数据传输部分中。
> - 下表中的第一列列出了您的AWS账单报告中显示的费用，并解释了每笔费用的含义。
> - 第二列列出了AWS使用情况报告中显示的项目，并显示了账单费用与使用情况报告项目之间的关联。

您的AWS账单中的 CloudFront 费用	AWS使用情况报告的 UsageType 列中的值
region-DataTransfer-Out-Bytes 为响应用户的 GET 和 HEAD 请求而从位于 *region* 的 CloudFront 边缘站点提供的总字节数。	*region*-Out-Bytes-HTTP-Static： 通过 HTTP 为 TTL ≥ 3600 秒的对象提供的字节数。 *region*-Out-Bytes-HTTPS-Static：

您的AWS账单中的 CloudFront 费用	AWS使用情况报告的 UsageType 列中的值
	通过 HTTPS 为 TTL ≥ 3600 秒的对象提供的字节数。 *region*-Out-Bytes-HTTP-Dynamic： 通过 HTTP 为 TTL < 3600 秒的对象提供的字节数。 *region*-Out-Bytes-HTTPS-Dynamic： 通过 HTTPS 为 TTL < 3600 秒的对象提供的字节数。 *region*-Out-Bytes-HTTP-Proxy： 为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTP 从 CloudFront 返回到查看器的字节数。 *region*-Out-Bytes-HTTPS-Proxy： 为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTPS 从 CloudFront 返回到查看器的字节数。
region-DataTransfer-Out-OBytes 为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而从 CloudFront 边缘站点向您的源或 Edge 函数传输的总字节数。费用包括将 WebSocket 数据从客户端传输到服务器的费用。	*region*-Out-OBytes-HTTP-Proxy 为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTP 从 CloudFront 边缘站点向您的源或 Edge 函数传输的总字节数。 *region*-Out-OBytes-HTTPS-Proxy 为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTPS 从 CloudFront 边缘站点向您的源或 Edge 函数传输的总字节数。

您的AWS账单中的 CloudFront 费用	AWS使用情况报告的 UsageType 列中的值
region-Requests-Tier1 HTTP GET 和 HEAD 请求的数量。	*region*-Requests-HTTP-Static 为 TTL ≥ 3600 秒的对象提供的 HTTP GET 和 HEAD 请求的数量。 *region*-Requests-HTTP-Dynamic 为 TTL < 3600 秒的对象提供的 HTTP GET 和 HEAD 请求的数量。
region-Requests-Tier2-HTTPS HTTPS GET 和 HEAD 请求的数量。	*region*-Requests-HTTPS-Static 为 TTL ≥ 3600 秒的对象提供的 HTTPS GET 和 HEAD 请求的数量。 *region*-Requests-HTTPS-Dynamic 为 TTL < 3600 秒的对象提供的 HTTPS GET 和 HEAD 请求的数量。
region-Requests-HTTP-Proxy CloudFront 转发到源或 Edge 函数的 HTTP DELETE、OPTIONS、PATCH、POST 和 PUT 请求的数量。 还包括 CloudFront 转发至源或边缘函数的 HTTP WebSocket 请求（包含 Upgrade: websocket 标头的 GET 请求）的数量。	*region*-Requests-HTTP-Proxy 与您的 CloudFront 账单中的相应项目相同。

您的AWS账单中的 CloudFront 费用	AWS使用情况报告的 UsageType 列中的值
region-Requests-HTTPS-Proxy CloudFront 转发到源或 Edge 函数的 HTTPS DELETE、OPTIONS、PATCH、POST 和 PUT 请求的数量。 还包括 CloudFront 转发至源或边缘函数的 HTTPS WebSocket 请求（包含 Upgrade: websocket 标头的 GET 请求）的数量。	*region*-Requests-HTTPS-Proxy 与您的 CloudFront 账单中的相应项目相同。
region-Requests-HTTPS-Proxy-FLE CloudFront 向您的源或 Edge 函数转发的且使用字段级加密处理的 HTTPS DELETE、OPTIONS、PATCH 和 POST 请求的数量。	*region*-Requests-HTTPS-Proxy-FLE 与您的 CloudFront 账单中的相应项目相同。
region-Bytes-OriginShield 从源传输至任何区域边缘缓存的总字节数，包括启用为 Origin Shield 的区域边缘缓存。	*region*-Bytes-OriginShield 与您的 CloudFront 账单中的相应项目相同。
region-OBytes-OriginShield 从任何区域边缘缓存传输至源的总字节数，包括启用为 Origin Shield 的区域边缘缓存。	*region*-OBytes-OriginShield 与您的 CloudFront 账单中的相应项目相同。
region-Requests-OriginShield 作为增量层传输至 Origin Shield 的请求数。对于通过代理到达源的动态（不可缓存）请求，Origin Shield 始终是增量层。对于可缓存请求，Origin Shield 有时是一个增量层。 有关更多信息，请参阅 the section called “估算 Origin Shield 成本”。	*region*-Requests-OriginShield 与您的 CloudFront 账单中的相应项目相同。

您的AWS账单中的 CloudFront 费用	AWS使用情况报告的 UsageType 列中的值
失效 使对象失效（将对象从 CloudFront 边缘站点中删除）所产生的费用。有关更多信息，请参阅支付文件失效费用。	失效 与您的 CloudFront 账单中的相应项目相同。
SSL-Cert-Custom 使用 SSL 证书及 CloudFront 备用域名（例如 example.com）而不是使用 CloudFront 为您的分配指定的默认 CloudFront SSL 证书及域名所产生的费用。	SSL-Cert-Custom 与您的 CloudFront 账单中的相应项目相同。
RealTimeLog-KinesisDataStream 按为实时日志生成的行数收取的费用。	RealTimeLog-KinesisDataStream 与您的 CloudFront 账单中的相应项目相同。
Executions-CloudFrontFunctions 按 CloudFront Functions 调用次数收取的费用。	Executions-CloudFrontFunctions 与您的 CloudFront 账单中的相应项目相同。
##-Lambda-Edge-Request 按 Lambda@Edge 函数调用次数收取的费用。	##-Lambda-Edge-Request 与您的 CloudFront 账单中的相应项目相同。
##-Lambda-Edge-GB-Second 从您的 Lambda@Edge 函数被调用到函数返回或终止的时间段内的费用。	##-Lambda-Edge-GB-Second 与您的 CloudFront 账单中的相应项目相同。
KeyValueStore-EdgeReads 按对 CloudFront KeyValueStore 方法、`get()`、`exists()` 和 `meta()` 的读取调用次数收取的费用。有关更多信息，请参阅 键值存储的帮助程序方法。	KeyValueStore-EdgeReads 与您的 CloudFront 账单中的相应项目相同。

您的AWS账单中的 CloudFront 费用	AWS使用情况报告的 UsageType 列中的值
KeyValueStore-APIOperations 按对 CloudFront KeyValueStore API 的调用次数收取的费用。	KeyValueStore-APIOperations 与您的 CloudFront 账单中的相应项目相同。

在控制台中查看 CloudFront 报告

您可以在控制台中查看有关您的 CloudFront 活动的以下报告：

主题

- 查看 CloudFront 缓存统计报告
- 查看 CloudFront 常用对象报告
- 查看 CloudFront 常用引用站点报告
- 查看 CloudFront 使用情况报告
- 查看 CloudFront 查看器报告

上述多数报告都是基于 CloudFront 访问日志中的数据，其中详细介绍了 CloudFront 收到的每个用户请求。您无需启用访问日志即可查看此类报告。有关更多信息，请参阅配置和使用标准日志（访问日志）。

查看 CloudFront 缓存统计报告

Amazon CloudFront 缓存统计报告包括以下信息：

- 请求总数 – 显示所有 HTTP 状态代码（例如，200 或 404）以及所有方法（例如，GET、HEAD 或 POST）的请求总数。
- 查看器请求所占的百分比(按结果类型划分) – 显示选定的 CloudFront 分配的命中数、未命中数和错误数占查看器总请求数的百分比。
- 传输到查看器的字节数 – 显示总字节数和未命中的字节数。
- HTTP 状态代码 – 显示按 HTTP 状态代码划分的查看器请求。
- 未完成下载的 GET 请求所占的百分比 – 显示未完成下载请求的对象的查看器 GET 请求数占总请求数的百分比。

这些统计数据与 CloudFront 访问日志来自同一来源，但您无需启用访问日志记录即可查看缓存统计信息。

您可以显示过去 60 天内指定日期范围的图表，将每小时或每天作为数据点。通常，您可以查看 CloudFront 在一小时前收到的最新请求的相关数据，但数据偶尔会延迟（长达 24 小时）。

主题

在控制台中查看 CloudFront 缓存统计报告

您可以在控制台中查看 CloudFront 缓存统计报告。

查看 CloudFront 缓存统计报告

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，单击缓存统计数据。
3. 在 CloudFront 缓存统计报告窗格中，为开始日期和结束日期选择要显示缓存统计图表的日期范围。可用的范围取决于您为粒度选择的值：

   - 每天 – 要显示每天一个数据点的图表，请选择前 60 天内的任何日期范围。
   - 每小时 – 要显示每小时一个数据点的图表，请选择前 60 天内的最多 14 天的任何日期范围。

   日期和时间采用协调世界时 (UTC)。
4. 对于粒度，请指定在图表中是每天显示一个数据点还是每小时显示一个数据点。如果您指定大于 14 天的日期范围，则指定每小时一个数据点的选项不可用。
5. 对于查看器位置，请选择查看器请求来自哪个洲，或选择所有位置。缓存统计信息表包括 CloudFront 收到的来自指定位置的请求数据。
6. 在分配列表中，选择您要在使用情况图表中显示数据的分配：

   - 单个分配 – 此类图表显示选定的 CloudFront Web 分配的数据。分配列表显示分配 ID 和分配的备用域名 (CNAME)（如果有）。如果某个分配没有备用域名，该列表将包含该分配的原始域名。

- 所有分配 – 此类图表显示与当前AWS账户关联的所有分配（不包括已删除的分配）的汇总数据。

7. 选择更新。

要查看图表内每日或每小时数据点的数据，请将鼠标指针悬停在相应数据点上。

对于显示所传输的数据的图表，请注意，您可以将每个图表的垂直比例更改为千兆字节、兆字节或千字节。

以 CSV 格式下载数据

您可以将缓存统计报告下载为 CSV 格式的文件。本部分将介绍如何下载该报告并详细说明其中的各个值。

以 CSV 格式下载缓存统计报告

1. 在查看缓存统计报告时，选择 CSV。
2. 在打开文件名对话框中，选择是要打开还是保存文件。

该报告的相关信息

该报告的前几行包含以下信息：

版本

该 CSV 文件格式的版本。

报告

报告的名称。

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

StartDateUTC

您运行报告的日期范围的开始日期，采用协调世界时 (UTC)。

EndDateUTC

您运行报告的日期范围的结束日期，采用协调世界时 (UTC)。

GeneratedTimeUTC

您运行报告的日期和时间，采用协调世界时 (UTC)。

粒度

报告中的每一行代表一小时还是一天。

ViewerLocation

查看器请求来自哪个洲或 ALL (如果您选择下载所有位置的报告)。

缓存统计信息报告中的数据

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

ViewerLocation

查看器请求来自哪个洲或 ALL (如果您选择下载所有位置的报告)。

TimeBucket

数据适用的小时或天，采用协调世界时 (UTC)。

RequestCount

所有 HTTP 状态代码 (例如，200 或 404) 及所有方法 (例如，GET、HEAD 或 POST) 的请求总数。

HitCount

从 CloudFront 边缘缓存内为其提供对象的查看器请求数。

MissCount

对象目前不在边缘缓存中，因此 CloudFront 必须从源获取对象的查看器请求数。

错误计数

导致错误，因此 CloudFront 未提供对象的查看器请求数。

IncompleteDownloadCount

查看器已启动但未完成对象下载的查看器请求数。

HTTP2xx

HTTP 状态代码为 2xx 值（已成功）的查看器请求数。

HTTP3xx

HTTP 状态代码为 3xx 值（要求执行其他操作）的查看器请求数。

HTTP4xx

HTTP 状态代码为 4xx 值（客户端错误）的查看器请求数。

HTTP5xx

HTTP 状态代码为 5xx 值 (服务器错误) 的查看器请求数。

TotalBytes

CloudFront 为响应所有 HTTP 方法的所有请求而提供给查看器的字节总数。

BytesFromMisses

对于在收到请求时不在边缘缓存中的对象，向查看器提供的字节数。该值与从源传输到 CloudFront 边缘缓存的字节数非常近似。但是，它不包括已存在于边缘缓存中但已过期的对象请求数。

缓存统计图与 CloudFront 标准日志（访问日志）中数据的相关度

下表显示了 CloudFront 控制台中的缓存统计图与 CloudFront 访问日志中值的对应情况。有关 CloudFront 访问日志的更多信息，请参阅配置和使用标准日志（访问日志）。

请求总数

该图表显示了所有 HTTP 状态代码 (例如，200 或 404) 及所有方法 (例如，GET、HEAD 或 POST) 的请求总数。该图表中显示的请求总数与同一时间段内访问日志文件的请求总数相等。

查看器请求所占的百分比（按结果类型划分）

该图表显示了所选 CloudFront 分配的命中数、未命中数和错误数占查看器请求总数的百分比：

- 命中 – 从 CloudFront 边缘缓存中为其提供对象的查看器请求。在访问日志中，此类请求的 `x-edge-response-result-type` 值为 `Hit`。
- 未命中 – 对象目前没有位于边缘缓存中，因此，CloudFront 必须从源中获取对象的查看器请求。在访问日志中，此类请求的 `x-edge-response-result-type` 值为 `Miss`。

- 错误 – 查看器请求导致错误，因此，CloudFront 未提供该对象。在访问日志中，此类请求的 x-edge-response-result-type 值为 Error、LimitExceeded 或 CapacityExceeded。

该图表不包括刷新命中 – 在边缘缓存中但已过期的对象的请求。在访问日志中，刷新命中是指 x-edge-response-result-type 值为 RefreshHit 的请求。

传输到查看器的字节数

该图表显示以下两个值：

- 总字节数 – CloudFront 为响应所有 HTTP 方法的所有请求而提供给查看器的总字节数。在 CloudFront 访问日志中，总字节数是同一时间段内所有请求的 sc-bytes 列中的值之和。
- 未命中的字节数 – 对于在收到请求时不在边缘缓存中的对象，向查看器提供的字节数。在 CloudFront 访问日志中，未命中的字节数是 sc-bytes 值为 x-edge-result-type 的请求的 Miss 列中的值之和。该值与从源传输到 CloudFront 边缘缓存的字节数非常近似。但是，它不包括已存在于边缘缓存中但已过期的对象请求数。

HTTP 状态代码

该图表显示了按 HTTP 状态代码划分的查看器请求。在 CloudFront 访问日志中，状态代码显示在 sc-status 列中：

- 2xx – 请求成功。
- 3xx – 需要执行其他操作。例如，301 (Moved Permanently) 表示请求的对象已移到其他位置。
- 4xx – 客户端明显出现错误。例如，404 (Not Found) 表示无法找到客户端请求的对象。
- 5xx – 源服务器未填充请求。例如，503 (Service Unavailable) 表示源服务器当前不可用。

未完成下载的 GET 请求所占的百分比

该图表显示了未完成所请求对象下载的查看器 GET 请求数占总请求数的百分比。通常，对象下载未完成是因为查看器取消了下载，例如，通过单击其他链接或关闭浏览器。在 CloudFront 访问日志中，此类请求在 200 列中具有 sc-status 值，在 Error 列中具有 x-edge-result-type 值。

查看 CloudFront 常用对象报告

查看 Amazon CloudFront 常用对象报告，以便查看在过去 60 天中指定日期范围内分配的 50 个最常用对象。您还可以查看有关这些对象的统计数据，包括以下信息：

- 对象的请求数
- 命中和未命中次数

- 命中率
- 为未命中提供的字节数
- 提供的总字节数
- 未完成下载次数
- 按 HTTP 状态代码（2xx、3xx、4xx 和 5xx）列出的请求数

这些统计数据与 CloudFront 访问日志来自同一来源，但您无需启用访问日志记录即可查看常用对象。

主题

在控制台中查看 CloudFront 常用对象报告

您可以在控制台中查看 CloudFront 常用对象报告。

查看 CloudFront 分配的常用对象

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，单击常用对象。
3. 在 CloudFront 常用对象报告窗格中，对于开始日期和结束日期，请选择您要显示常用对象列表的日期范围。您可以选择前 60 天内的任何日期范围。

   日期和时间采用协调世界时 (UTC)。
4. 在分配列表中，选择要显示常用对象列表的分配。
5. 选择更新。

CloudFront 如何计算常用对象统计数据

为了获取您的分配中前 50 个对象的准确计数，CloudFront 会从午夜开始，以 10 分钟为间隔来统计所有对象的请求，并在接下来 24 小时不间断地汇总前 150 个对象。（CloudFront 还会保留前 150 个对象在 60 天内的每日总计。）

在列表底部附近，不断有对象进入或跌出该列表，因此，这些对象的总计为近似值。在 150 个对象列表中，前面的 50 个对象可能会在列表内上升或下降，但很少会跌出列表，因此，这些对象的总计通常会更可靠。

如果某个对象在一天内跌出前 150 个对象列表后又重新回到列表中，CloudFront 会添加该对象从列表中消失的那段时间内估计的请求数。该估计值基于列表底部的任一对象在该时间段内收到的请求数。

如果该对象在当天稍晚时候进入了前 50 个对象之列，则在其尚未进入前 150 个对象之列时，CloudFront 收到的估计请求数通常会导致常用对象报告中的请求数超过该对象在访问日志中显示的请求数。

以 CSV 格式下载数据

您能够以 CSV 格式下载常用对象报告。本部分将介绍如何下载该报告并详细说明其中的各个值。

以 CSV 格式下载常用对象报告

1. 在查看常用对象报告时，选择 CSV。
2. 在打开文件名对话框中，选择是要打开还是保存文件。

该报告的相关信息

该报告的前几行包含以下信息：

版本

该 CSV 文件格式的版本。

报告

报告的名称。

DistributionID

您运行报告的分配 ID。

StartDateUTC

您运行报告的日期范围的开始日期，采用协调世界时 (UTC)。

EndDateUTC

您运行报告的日期范围的结束日期，采用协调世界时 (UTC)。

GeneratedTimeUTC

您运行报告的日期和时间，采用协调世界时 (UTC)。

常用对象报告中的数据

该报告包括以下值：

DistributionID

您运行报告的分配 ID。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

对象

对象 URL 中的最后 500 个字符。

RequestCount

该对象的请求总数。

HitCount

从 CloudFront 边缘缓存内为其提供对象的查看器请求数。

MissCount

对象目前不在边缘缓存中，因此 CloudFront 必须从源获取对象的查看器请求数。

HitCountPct

`HitCount` 值占 `RequestCount` 值的百分比。

BytesFromMisses

系统为响应不在边缘缓存中的这类对象请求而提供给查看器的字节数。

TotalBytes

CloudFront 为响应所有 HTTP 方法的所有此类对象请求而提供给查看器的字节总数。

IncompleteDownloadCount

查看器已启动但未完成对象下载的此类对象的查看器请求数。

HTTP2xx

HTTP 状态代码为 2xx 值（已成功）的查看器请求数。

HTTP3xx

HTTP 状态代码为 3xx 值（要求执行其他操作）的查看器请求数。

HTTP4xx

HTTP 状态代码为 4xx 值（客户端错误）的查看器请求数。

HTTP5xx

HTTP 状态代码为 5xx 值（服务器错误）的查看器请求数。

常用对象报告中的数据与 CloudFront 标准日志（访问日志）中数据的相关度

以下列表显示了 CloudFront 控制台中常用对象报告的值与 CloudFront 访问日志中值的对应情况。有关 CloudFront 访问日志的更多信息，请参阅配置和使用标准日志（访问日志）。

URL

查看器用来访问对象的 URL 中的最后 500 个字符。

请求

对象的请求总数。该值通常与 CloudFront 访问日志中对象的 GET 请求数密切对应。

命中数

从 CloudFront 边缘缓存提供对象的查看器请求数。在访问日志中，此类请求的 x-edge-response-result-type 值为 Hit。

未命中数

对象不在边缘缓存中，因此 CloudFront 从源中检索了对象的查看器请求数。在访问日志中，此类请求的 x-edge-response-result-type 值为 Miss。

命中率

Hits 列中的值占 Requests 列中值的百分比。

未命中的字节数

对于在收到请求时不在边缘缓存中的对象，向查看器提供的字节数。在 CloudFront 访问日志中，未命中的字节数是 sc-bytes 值为 x-edge-result-type 的请求的 Miss 列中的值之和。

总字节数

CloudFront 为响应所有 HTTP 方法的所有对象请求而提供给查看器的总字节数。在 CloudFront 访问日志中，总字节数是同一时间段内所有请求的 `sc-bytes` 列中的值之和。

未完成的下载数量

未完成所请求对象下载的查看器请求数。通常，下载未完成是因为查看器取消了下载，例如，通过单击其他链接或关闭浏览器。在 CloudFront 访问日志中，此类请求在 `200` 列中具有 `sc-status` 值，在 `Error` 列中具有 `x-edge-result-type` 值。

2xx

HTTP 状态代码为 `2xx`、`Successful` 的请求数。在 CloudFront 访问日志中，状态代码显示在 `sc-status` 列中。

3xx

HTTP 状态代码为 `3xx Redirection` 的请求数。3xx 状态代码表示需要执行其他操作。例如，301 (Moved Permanently) 表示请求的对象已移到其他位置。

4xx

HTTP 状态代码为 `4xx Client Error` 的请求数。4xx 状态代码表示客户端明显出现错误。例如，404 (Not Found) 表示无法找到客户端请求的对象。

5xx

HTTP 状态代码为 `5xx Server Error` 的请求数。5xx 状态代码表示源服务器未填充请求。例如，503 (Service Unavailable) 表示源服务器当前不可用。

查看 CloudFront 常用引用站点报告

CloudFront 常用引用站点报告包含过去 60 天内任何日期范围内的以下信息：

- 前 25 个引用站点（针对 CloudFront 为您的分配分发的对象，发起最多 HTTP和 HTTPS 请求的网站域）
- 来自引用站点的请求数
- 引用站点在指定时间段内提交的请求数占总请求数的百分比

常用引用站点报告数据与 CloudFront 访问日志来自同一来源，但您无需启用访问日志记录即可查看常用引用站点。

常用引用站点可以是搜索引擎、直接链接到您的对象的其他网站或者您自己的网站。例如，如果 `https://example.com/index.html` 链接到 10 个图形，则 `example.com` 是所有 10 个图形的引用站点。

> **Note**
>
> 如果用户直接在浏览器地址栏中输入 URL，则所请求的对象就不存在引用站点。

主题

在控制台中查看 CloudFront 常用引用站点报告

您可以在控制台中查看 CloudFront 常用引用站点报告。

查看 CloudFront 分配的常用引用站点

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择常用引用站点。
3. 在 CloudFront 常用引用站点报告窗格中，对于开始日期和结束日期，请选择您要显示常用引用站点列表的日期范围。

   日期和时间采用协调世界时 (UTC)。
4. 在分配列表中，选择要显示常用引用站点列表的分配。
5. 选择更新。

CloudFront 如何计算常用引用站点统计数据

为了获取前 25 个引用站点的准确计数，CloudFront 会以 10 分钟为间隔来统计所有对象的请求，并维持前 75 个引用站点的汇总。在列表底部附近，不断有引用站点进入或跌出该列表，因此，这些引用站点的总数为近似值。

在 75 个引用站点列表中，前面的 25 个引用站点可能会在列表内上升或下降，但很少会跌出列表，因此，这些引用站点的总计通常会更可靠。

以 CSV 格式下载数据

您可以将常用引用站点报告下载为 CSV 格式的文件。本部分将介绍如何下载该报告并详细说明其中的各个值。

将常用引用站点报告下载为 CSV 格式的文件

1. 在查看常用引用站点报告时，选择 CSV。
2. 在打开文件名对话框中，选择是要打开还是保存文件。

该报告的相关信息

该报告的前几行包含以下信息：

版本

该 CSV 文件格式的版本。

报告

报告的名称。

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

StartDateUTC

您运行报告的日期范围的开始日期，采用协调世界时 (UTC)。

EndDateUTC

您运行报告的日期范围的结束日期，采用协调世界时 (UTC)。

GeneratedTimeUTC

您运行报告的日期和时间，采用协调世界时 (UTC)。

常用引用站点报告中的数据

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

引用站点

引用站点的域名。

RequestCount

从 Referrer 列中的域名发出的请求总数。

RequestsPct

引用站点在指定时间段内提交的请求数占总请求数的百分比。

常用引用站点报告中的数据与 CloudFront 标准日志（访问日志）中数据的相关度

以下列表显示了 CloudFront 控制台中常用引用站点报告的值与 CloudFront 访问日志中值的对应情况。有关 CloudFront 访问日志的更多信息，请参阅配置和使用标准日志（访问日志）。

引用站点

引用站点的域名。在访问日志中，引用站点显示在 cs(Referer) 列中。

请求计数

从 Referrer 列中的域名发出的请求总数。该值通常与 CloudFront 访问日志中引用站点的 GET 请求数密切对应。

请求 %

引用站点在指定时间段内提交的请求数占总请求数的百分比。如果您的引用站点超过 25 个，则您无法根据本表中的数据计算请求 %，因为请求计数列不包括指定时间段内的所有请求数。

查看 CloudFront 使用情况报告

CloudFront 使用情况报告包括以下信息：

- 请求数 – 显示在指定的 CloudFront 分配的每个时间间隔内，CloudFront 响应来自选定区域中的边缘站点的请求总数。
- 按协议划分的已传输数据和按目的地划分的已传输数据 – 显示在指定的 CloudFront 分配的每个时间间隔内，从选定区域中的 CloudFront 边缘站点传输的数据总量。它们以不同的方式分隔数据，如下所示：
  - 按协议 – 按协议分隔数据：HTTP 或 HTTPS。
  - 按目的地 – 按目的地分隔数据：分隔到您的查看器或源。

CloudFront 使用情况报告基于 CloudFront 提供的 AWS 使用情况报告，它也无需任何特殊配置。有关更多信息，请参阅 查看 CloudFront 的AWS使用情况报告。

您可以查看过去 60 天内指定日期范围的图表，将每小时或每天作为数据点。通常，您可以查看 CloudFront 在 4 小时前收到的最新请求的相关数据，但数据偶尔会延迟（长达 24 小时）。

有关更多信息，请参阅 使用率图表与 CloudFront 使用率报告中的数据的相关度。

主题

在控制台中查看 CloudFront 使用情况报告

您可以在控制台中查看 CloudFront 使用情况报告。

查看 CloudFront 使用情况报告

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择使用情况报告。
3. 在 CloudFront 使用情况报告窗格中，对于开始日期和结束日期，请选择您要显示使用情况图表的日期范围。可用的范围取决于您为粒度选择的值：
   - 每天 – 要显示每天一个数据点的图表，请选择前 60 天内的任何日期范围。
   - 每小时 – 要显示每小时一个数据点的图表，请选择前 60 天内的最多 14 天的任何日期范围。

日期和时间采用协调世界时 (UTC)。

4. 对于粒度，请指定在图表中是每天显示一个数据点还是每小时显示一个数据点。如果您指定大于 14 天的日期范围，则指定每小时一个数据点的选项不可用。
5. 对于账单区域，请选择包含您要查看的数据的 CloudFront 账单区域，或选择所有区域。使用情况图表包含 CloudFront 处理的指定区域中边缘站点的请求数据。CloudFront 处理请求的区域不一定与您的查看器的位置对应。

   请仅选择包含在分配的价格级别中的区域。否则，使用情况图表可能不会包含任何数据。例如，如果您为分配选择了价格级别 200，则不会包含南美洲和澳大利亚账单区域，因此 CloudFront 一般不会处理您来自这些区域的请求。有关定价级别的更多信息，请参阅 CloudFront 定价。
6. 在分配列表中，选择您要在使用情况图表中显示数据的分配：

   - 单个分配 — 此类图表显示选定的 CloudFront Web 分配的数据。分配列表显示分配 ID 和分配的备用域名 (CNAME)（如果有）。如果某个分配没有备用域名，该列表将包含该分配的原始域名。
   - 所有分配（不含已删除的分配）– 此类图表显示与当前AWS账户关联的所有分配（不包括您已删除的分配）的汇总数据。
   - 所有删除的分配 – 此类图表显示与当前AWS账户关联但已在过去 60 天内删除的所有分配的汇总数据。
7. 选择更新图表。

要查看图表内每日或每小时数据点的数据，请将鼠标指针悬停在相应数据点上。

对于显示所传输的数据的图表，请注意，您可以将每个图表的垂直比例更改为千兆字节、兆字节或千字节。

以 CSV 格式下载数据

您可以下载 CSV 格式的使用情况报告。本部分将介绍如何下载该报告并详细说明其中的各个值。

下载 CSV 格式的使用情况报告

1. 在查看使用情况报告时，选择 CSV。
2. 在打开文件名对话框中，选择是要打开还是保存文件。

该报告的相关信息

该报告的前几行包含以下信息：

版本

该 CSV 文件格式的版本。

报告

报告的名称。

DistributionID

您运行报告的分配 ID、ALL (如果您运行了所有分配的报告) 或 ALL_DELETED (如果您运行了所有已删除的分配的报告)。

StartDateUTC

您运行报告的日期范围的开始日期，采用协调世界时 (UTC)。

EndDateUTC

您运行报告的日期范围的结束日期，采用协调世界时 (UTC)。

GeneratedTimeUTC

您运行报告的日期和时间，采用协调世界时 (UTC)。

粒度

报告中的每一行代表一小时还是一天。

BillingRegion

查看器请求来自哪个洲或 ALL (如果您选择下载所有账单区域的报告)。

使用情况报告中的数据

该报告包括以下值：

DistributionID

您运行报告的分配 ID、ALL (如果您运行了所有分配的报告) 或 ALL_DELETED (如果您运行了所有已删除的分配的报告)。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

BillingRegion

您运行报告的 CloudFront 账单区域或 ALL。

TimeBucket

数据适用的小时或天，采用协调世界时 (UTC)。

HTTP

在指定 CloudFront 分配的每个时间间隔内，CloudFront 响应从所选区域中边缘站点发出的 HTTP 请求数。值包括：

- GET 和 HEAD 请求数，这些请求促使 CloudFront 将数据传输给您的查看器
- DELETE、OPTIONS、PATCH、POST 和 PUT 请求数，这些请求促使 CloudFront 将数据传输给您的源

HTTPS

在指定 CloudFront 分配的每个时间间隔内，CloudFront 响应从所选区域中边缘站点发出的 HTTPS 请求数。值包括：

- GET 和 HEAD 请求数，这些请求促使 CloudFront 将数据传输给您的查看器
- DELETE、OPTIONS、PATCH、POST 和 PUT 请求数，这些请求促使 CloudFront 将数据传输给您的源

HTTPBytes

在指定 CloudFront 分配的时间段内，从所选账单区域中的 CloudFront 边缘站点通过 HTTP 传输的数据总量。值包括：

- 为响应 GET 和 HEAD 请求而从 CloudFront 传输到查看器的数据
- DELETE、OPTIONS、PATCH、POST 和 PUT 请求从查看器传输到 CloudFront 的数据
- 为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而从 CloudFront 传输到查看器的数据

HTTPSBytes

在指定 CloudFront 分配的时间段内，从所选账单区域中的 CloudFront 边缘站点通过 HTTPS 传输的数据总量。值包括：

- 为响应 GET 和 HEAD 请求而从 CloudFront 传输到查看器的数据

- DELETE、OPTIONS、PATCH、POST 和 PUT 请求从查看器传输到 CloudFront 的数据
- 为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而从 CloudFront 传输到查看器的数据

BytesIn

在指定 CloudFront 分配的每个时间间隔内，为响应所选区域中的 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而从 CloudFront 传输给源的数据总量。

BytesOut

在指定 CloudFront 分配的每个时间间隔内，通过 HTTP 和 HTTPS 在所选区域中从 CloudFront 传输给查看器的数据总量。值包括：

- 为响应 GET 和 HEAD 请求而从 CloudFront 传输到查看器的数据
- 为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而从 CloudFront 传输到查看器的数据

使用率图表与 CloudFront 使用率报告中的数据的相关度

以下列表显示了 CloudFront 控制台中的使用率图表与 CloudFront 使用率报告中使用类型列内各值的对应情况。

主题

- 请求数
- 按协议划分的已传输数据
- 按目的地划分的已传输数据

请求数

此图表显示了在指定的 CloudFront 分配的每个时间间隔期间从所选区域的边缘站点 CloudFront 响应的请求总数，这些请求按协议（HTTP 或 HTTPS）和类型（静态、动态或代理）分隔。

HTTP 请求的数量

- *region*-Requests-HTTP-Static：为 TTL ≥ 3600 秒的对象提供的 HTTP GET 和 HEAD 请求数
- *region*-Requests-HTTP-Dynamic：为 TTL < 3600 秒的对象提供的 HTTP GET 和 HEAD 请求数
- *region*-Requests-HTTP-Proxy：CloudFront 转发到源的 HTTP DELETE、OPTIONS、PATCH、POST 和 PUT 请求数

HTTPS 请求的数量

- *region*-Requests-HTTPS-Static：为 TTL ≥ 3600 秒的对象提供的 HTTPS GET 和 HEAD 请求数

- *region*-Requests-HTTPS-Dynamic：为 TTL < 3600 秒的对象提供的 HTTPS GET 和 HEAD 请求数
- *region*-Requests-HTTPS-Proxy：CloudFront 转发到源的 HTTPS DELETE、OPTIONS、PATCH、POST 和 PUT 请求数

按协议划分的已传输数据

此图表显示在指定的 CloudFront 分配的每个时间间隔期间，从所选区域的 CloudFront 边缘站点传输的数据总量，这些数据按协议（HTTP 或 HTTPS）、类型（静态、动态或代理）和目的地（查看器或源）分隔。

通过 HTTP 传输的数据

- *region*-Out-Bytes-HTTP-Static：通过 HTTP 为 TTL ≥ 3600 秒的对象提供的字节数
- *region*-Out-Bytes-HTTP-Dynamic：通过 HTTP 为 TTL < 3600 秒的对象提供的字节数
- *region*-Out-Bytes-HTTP-Proxy：为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTP 从 CloudFront 返回到查看器的字节数
- *region*-Out-OBytes-HTTP-Proxy：为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTP 从 CloudFront 边缘站点传输到源的字节总数

通过 HTTPS 传输的数据

- *region*-Out-Bytes-HTTPS-Static：通过 HTTPS 为 TTL ≥ 3600 秒的对象提供的字节数
- *region*-Out-Bytes-HTTPS-Dynamic：通过 HTTPS 为 TTL < 3600 秒的对象提供的字节数
- *region*-Out-Bytes-HTTPS-Proxy：为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTPS 从 CloudFront 返回到查看器的字节数
- *region*-Out-OBytes-HTTPS-Proxy：为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTPS 从 CloudFront 边缘站点传输到源的字节总数

按目的地划分的已传输数据

此图表显示在指定的 CloudFront 分配的每个时间间隔期间，从所选区域的 CloudFront 边缘站点传输的数据总量，这些数据按目的地（查看器或源）、协议（HTTP 或 HTTPS）和类型（静态、动态或代理）分隔。

从 CloudFront 传输到您的查看器的数据

- *region*-Out-Bytes-HTTP-Static：通过 HTTP 为 TTL ≥ 3600 秒的对象提供的字节数
- *region*-Out-Bytes-HTTPS-Static：通过 HTTPS 为 TTL ≥ 3600 秒的对象提供的字节数

- *region*-Out-Bytes-HTTP-Dynamic：通过 HTTP 为 TTL < 3600 秒的对象提供的字节数
- *region*-Out-Bytes-HTTPS-Dynamic：通过 HTTPS 为 TTL < 3600 秒的对象提供的字节数
- *region*-Out-Bytes-HTTP-Proxy：为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTP 从 CloudFront 返回到查看器的字节数
- *region*-Out-Bytes-HTTPS-Proxy：为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTPS 从 CloudFront 返回到查看器的字节数

从 CloudFront 传输到您的源的数据

- *region*-Out-OBytes-HTTP-Proxy：为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTP 从 CloudFront 边缘站点传输到源的字节总数
- *region*-Out-OBytes-HTTPS-Proxy：为响应 DELETE、OPTIONS、PATCH、POST 和 PUT 请求而通过 HTTPS 从 CloudFront 边缘站点传输到源的字节总数

查看 CloudFront 查看器报告

CloudFront 查看器报告包含过去 60 天内任何日期范围的以下信息：

- 设备 – 最常用于访问内容的设备类型（例如台式机或移动设备）
- 浏览器 – 最常用于访问您的内容的十大浏览器（例如 Chrome 或 Firefox）
- 操作系统 – 访问内容时最常使用的十大操作系统（例如 Linux、macOS 或 Windows）
- 位置 – 最常访问您的内容的查看器所在的前 50 个位置（国家/地区或省市/地区）。
  - 还可以查看过去 60 天内最多 14 天任何日期范围的每小时数据点的位置

您无需启用访问日志记录即可查看查看器图表和报告。

主题

在控制台中查看查看器图表和报告

您可以在控制台中查看 CloudFront 查看器图表和报告。

查看 CloudFront 查看器图表和报告

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 在导航窗格中，选择查看器。
3. 在 CloudFront 查看器窗格中，对于开始日期和结束日期，请选择您要显示查看器图表和报告的日期范围。

   对于位置图表，可用范围取决于您为粒度选择的值：

   - 每天 – 要显示每天一个数据点的图表，请选择前 60 天内的任何日期范围。
   - 每小时 – 要显示每小时一个数据点的图表，请选择前 60 天内的最多 14 天的任何日期范围。

   日期和时间采用协调世界时 (UTC)。
4. （仅限于浏览器和操作系统图表）对于分组，请指定您是要按名称 (Chrome、Firefox) 还是按名称和版本 (Chrome 40.0、Firefox 35.0) 对浏览器及操作系统分组。
5. （仅限于位置图表）对于粒度，请指定在图表中是将每天显示为一个数据点还是将每小时显示为一个数据点。如果您指定大于 14 天的日期范围，则指定每小时一个数据点的选项不可用。
6. （仅限于位置图表）对于详细信息，请指定是按国家/地区还是按美国各州来显示主要位置。
7. 在分配列表中，选择您要在使用情况图表中显示数据的分配：

   - 单个分配 – 此类图表显示选定的 CloudFront Web 分配的数据。分配列表显示分配 ID 和分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。
   - 所有分配（不含已删除的分配） – 此类图表显示与当前AWS账户关联的所有分配（不包括您已删除的分配）的汇总数据。
8. 选择更新。

要查看图表内每日或每小时数据点的数据，请将鼠标指针悬停在相应数据点上。

以 CSV 格式下载数据

您可以将每个查看器报告下载为 CSV 格式的文件。本部分将介绍如何下载此类报告并详细说明其中的各个值。

将查看器报告下载为 CSV 格式的文件

1. 在查看查看器报告时，选择 CSV。

2. 选择您要下载的数据，例如，Devices 或 Devices Trends。
3. 在打开 文件名对话框中，选择是要打开还是保存文件。

查看器报告中包含的数据

每个报告的前几行都包含以下信息：

版本

该 CSV 文件格式的版本。

报告

报告的名称。

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

StartDateUTC

您运行报告的日期范围的开始日期，采用协调世界时 (UTC)。

EndDateUTC

您运行报告的日期范围的结束日期，采用协调世界时 (UTC)。

GeneratedTimeUTC

您运行报告的日期和时间，采用协调世界时 (UTC)。

分组（仅限于浏览器和操作系统报告）

是按浏览器或操作系统的名称还是按名称和版本对数据分组。

粒度

报告中的每一行代表一小时还是一天。

详细信息（仅限于位置报告）

是按国家/地区还是按美国各州来列出请求。

以下主题介绍不同查看器报告中的信息。

主题

设备报告

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

请求

CloudFront 从各类设备收到的请求数。

RequestsPct

CloudFront 从各类设备收到的请求数占 CloudFront 从所有设备收到的总请求数的百分比。

设备趋势报告

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

TimeBucket

数据适用的小时或天，采用协调世界时 (UTC)。

桌面

CloudFront 在相应时间段内从台式计算机收到的请求数。

移动

CloudFront 在相应时间段内从移动设备收到的请求数。移动设备可以同时包括平板电脑和手机。如果 CloudFront 无法确定请求是来自移动设备还是平板电脑，则会将此类请求显示在 `Mobile` 列中。

智能电视

CloudFront 在相应时间段内从智能电视收到的请求数。

平板电脑

CloudFront 在相应时间段内从平板电脑收到的请求数。如果 CloudFront 无法确定请求是来自移动设备还是平板电脑，则会将此类请求显示在 `Mobile` 列中。

Unknown

`User-Agent` HTTP 标头值未与其中一个标准设备类型（例如，`Desktop` 或 `Mobile`）关联的请求。

Empty

CloudFront 在相应时间段内收到但不包含 HTTP `User-Agent` 标头值的请求数。

浏览器报告

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 `ALL`（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

组

CloudFront 从中收到请求的浏览器或浏览器和版本，具体取决于 Grouping 值。除了浏览器名称外，可能的值还包括以下各项：

- 自动程序/爬网程序 – 主要是来自搜索引擎且将内容编入索引的请求。
- 空 – User-Agent HTTP 标头值为空的请求。
- 其他 – CloudFront 已识别但不属于常见浏览器的浏览器。如果 Bot/Crawler、Empty 和/或 Unknown 不在前 9 个值之列，则系统还会将其显示在 Other 中。
- 未知 – User-Agent HTTP 标头值与标准浏览器不关联的请求。该类别中的大多数请求来自自定义应用程序或脚本。

请求

CloudFront 从各类浏览器收到的请求数。

RequestsPct

CloudFront 在相应时间段内从各类浏览器收到的请求数占 CloudFront 收到的总请求数的百分比。

浏览器趋势报告

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

TimeBucket

数据适用的小时或天，采用协调世界时 (UTC)。

(浏览器)

该报告中的其余列会列出浏览器或浏览器及其版本，具体取决于 Grouping 的值。除了浏览器名称外，可能的值还包括以下各项：

- 自动程序/爬网程序 – 主要是来自搜索引擎且将内容编入索引的请求。
- 空 – User-Agent HTTP 标头值为空的请求。
- 其他 – CloudFront 已识别但不属于常见浏览器的浏览器。如果 Bot/Crawler、Empty 和/或 Unknown 不在前 9 个值之列，则系统还会将其显示在 Other 中。
- 未知 – User-Agent HTTP 标头值与标准浏览器不关联的请求。该类别中的大多数请求来自自定义应用程序或脚本。

操作系统报告

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

组

CloudFront 从中收到请求的操作系统或操作系统和版本，具体取决于 Grouping 值。除了操作系统名称外，可能的值还包括以下各项：

- 自动程序/爬网程序 – 主要是来自搜索引擎且将内容编入索引的请求。
- 空 – User-Agent HTTP 标头值为空的请求。
- 其他 – CloudFront 已识别但不属于常见浏览器的操作系统。如果 Bot/Crawler、Empty 和/或 Unknown 不在前 9 个值之列，则系统还会将其显示在 Other 中。
- 未知 – User-Agent HTTP 标头值与标准浏览器不关联的请求。该类别中的大多数请求来自自定义应用程序或脚本。

请求

CloudFront 从各类操作系统收到的请求数。

RequestsPct

CloudFront 在相应时间段内从各类操作系统收到的请求数占 CloudFront 收到的总请求数的百分比。

操作系统趋势报告

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

TimeBucket

数据适用的小时或天，采用协调世界时 (UTC)。

(操作系统)

该报告中的其余列会列出操作系统或操作系统及其版本，具体取决于 Grouping 的值。除了操作系统名称外，可能的值还包括以下各项：

- 自动程序/爬网程序 – 主要是来自搜索引擎且将内容编入索引的请求。
- 空 – User-Agent HTTP 标头值为空的请求。
- 其他 – CloudFront 已识别但不属于常见浏览器的操作系统。如果 Bot/Crawler、Empty 和/或 Unknown 不在前 9 个值之列，则系统还会将其显示在 Other 中。
- 未知 – 未在 User-Agent HTTP 标头中指定操作系统的请求。

位置报告

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

LocationCode

CloudFront 从其收到请求的位置缩写。有关可能值的更多信息，请参阅 位置报告中的数据与 CloudFront 标准日志（访问日志）中数据的相关度 中的位置描述。

LocationName

CloudFront 从其收到请求的位置名称。

请求

CloudFront 从每个位置收到的请求数。

RequestsPct

CloudFront 在相应时间段内从各位置收到的请求数占 CloudFront 从所有位置收到的总请求数的百分比。

TotalBytes

针对指定的分配和时间段，CloudFront 提供给该国家/地区或州的查看器的字节数。

位置趋势报告

该报告包括以下值：

DistributionID

您运行报告的分配 ID 或 ALL（如果您运行了所有分配的报告）。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

TimeBucket

数据适用的小时或天，采用协调世界时 (UTC)。

(位置)

该报告中的剩余列会列出 CloudFront 从其收到请求的位置。有关可能值的更多信息，请参阅 位置报告中的数据与 CloudFront 标准日志（访问日志）中数据的相关度 中的位置描述。

位置报告中的数据与 CloudFront 标准日志（访问日志）中数据的相关度

以下列表显示了 CloudFront 控制台中位置报告的数据与 CloudFront 访问日志中值的对应情况。有关 CloudFront 访问日志的更多信息，请参阅配置和使用标准日志（访问日志）。

位置

查看器所在的国家/地区或美国的州。在访问日志中，c-ip 列包含运行查看器的设备的 IP 地址。我们使用地理位置数据基于 IP 地址识别设备的地理位置。

如果您显示按国家/地区划分的 Locations 报告，请注意，国家/地区列表基于 ISO 3166-2，即国家及其区域名表示代码 – 第 2 部分：国家区域代码。国家/地区列表包括以下其他值：

- 匿名代理 – 请求来自匿名代理。
- 卫星提供商 – 请求来自向多个国家/地区提供 Internet 服务的卫星提供商。查看器可能位于欺诈风险较高的国家/地区。
- 欧洲（未知）– 请求来自多个欧洲国家/地区使用的块中的 IP 地址。无法确定请求来自哪个国家/地区。CloudFront 将欧洲（未知）用作默认值。
- 亚太区域（未知）– 请求来自多个亚太国家/地区使用的块中的 IP 地址。无法确定请求来自哪个国家/地区。CloudFront 将亚太地区（未知）用作默认值。

如果您显示按美国各州划分的位置报告，请注意，该报告会包括美国领土和美国军事地区。

> **Note**
>
> 如果 CloudFront 无法确定某个用户的位置，该位置将在查看器报告中显示为未知。

请求计数

在指定分配的时间段内，来自查看器所在的国家/地区或美国各州的请求总数。该值通常与 CloudFront 访问日志中来自该国家/地区或州的 IP 地址的 GET 请求数密切对应。

请求 %

以下项之一，具体取决于您为详细信息选择的值：

- 国家/地区 – 来自该国家/地区的请求数占总请求数的百分比。
- 美国各州 – 来自该州的请求数占来自整个美国的请求总数的百分比。

如果请求来源超过 50 个国家/地区，则您无法根据本表中的数据计算请求 %，因为请求计数列不包括指定时间段内的所有请求数。

字节

针对指定的分配和时间段，CloudFront 提供给该国家/地区或州的查看器的字节数。要将该列中显示的数据单位更改为 KB、MB 或 GB，请单击列标题中的链接。

使用 Amazon CloudWatch 监控 CloudFront 指标

Amazon CloudFront 已与 Amazon CloudWatch 集成，可自动发布分配和边缘函数（Lambda@Edge 和 CloudFront）的运行指标。其中许多指标显示在CloudFront 控制台的一组图表中，还可以使用 CloudFront API 或 CLI 进行访问。所有这些指标均可通过 CloudWatch 控制台或者通过 CloudWatch API 或 CLI 进行访问。这些 CloudFront 指标不会计入 CloudWatch 配额（以前称为限制），也不会产生任何额外费用。

除了 CloudFront 分配的默认指标外，您还可以启用其他指标，但需要支付额外费用。其他指标适用于 CloudFront 分配，并且必须单独为每个分配启用。有关成本的更多信息，请参阅the section called “估算其他 CloudFront 指标的成本”。

查看这些指标可帮助您解决、跟踪和调试问题。要在 CloudFront 控制台中查看这些指标，请参阅监控页面。要查看有关特定 CloudFront 分配或边缘函数的活动的图表，请选择一项，然后选择 View distribution metrics（查看分配指标）或 View metrics（查看指标）。

您还可以在 CloudFront 控制台或 CloudWatch 控制台、API 或 CLI 中基于这些指标设置警报（适用标准 CloudWatch Logs 定价）。例如，您可以根据 `5xxErrorRate` 指标设置警报，该指标表示响应的 HTTP 状态代码在 `500` 到 `599`（含这两个值）范围内的所有查看器请求的百分比。当错误率在一定时间内达到某个值（例如，连续 5 分钟的请求的 5%）时，将触发警报。您可以在创建警报时指定警报的值及其时间单位。有关更多信息，请参阅 创建警报。

> **Note**
>
> 在 CloudFront 控制台中创建 CloudWatch 警报时，它将在美国东部（弗吉尼亚州北部）区域（`us-east-1`）为您创建一个警报。如果您通过 CloudWatch 控制台创建警报，则必须使用相同的区域。由于 CloudFront 是一项全球服务，因此该服务的相关指标会发送到美国东部（弗吉尼亚州北部）。

主题

- 查看 CloudFront 和边缘函数指标
- 为指标创建警报
- 以 CSV 格式下载指标数据
- 使用 CloudWatch API 获取指标

查看 CloudFront 和边缘函数指标

您可以在 CloudFront 控制台中查看有关您的 CloudFront 分配和边缘函数的运行指标。要查看这些指标，请参阅 CloudFront 控制台中的监控页面。要查看有关特定 CloudFront 分配或边缘函数的活动的图表，请选择一项，然后选择 View distribution metrics（查看分配指标）或 View metrics（查看指标）。

主题

查看默认的 CloudFront 分配指标

所有 CloudFront 分配都包含以下默认指标，无需额外费用：

请求

针对所有 HTTP 方法以及 HTTP 和 HTTPS 请求，CloudFront 收到的查看器请求总数。

已下载字节

查看器针对 GET、HEAD 和 OPTIONS 请求下载的字节总数。

已上传字节

查看器使用 POST 和 PUT 请求上传到 CloudFront 的字节总数。

4xx 错误率

响应的 HTTP 状态代码为 4xx 的所有查看器请求所占的百分比。

5xx 错误率

响应的 HTTP 状态代码为 5xx 的所有查看器请求所占的百分比。

总错误率

响应的 HTTP 状态代码为 4xx 或 5xx 的所有查看器请求所占的百分比。

在 CloudFront 控制台的监控页面上，每个 CloudFront 分配都会以图表形式显示这些指标。在每个图表上，总计值按 1 分钟的粒度显示。除了查看图表外，您还可以将指标报告下载为 CSV 文件。

您可以通过执行以下操作来自定义图表：

- 要更改图表中所显示信息的时间范围，请选择 1h（1 小时）、3h（3 小时）或其他范围，或指定自定义范围。
- 要更改 CloudFront 更新图表中信息的频率，请选择刷新图标旁边的下箭头，然后选择一个刷新间隔。默认刷新频率为 1 分钟，不过您可以选择 10 秒、2 分钟或其他选项。

要在 CloudWatch 控制台中查看 CloudFront 图形，请选择添加到控制面板。

启用其他 CloudFront 分配指标

除了默认指标外，您还可以启用其他指标，但需要支付额外费用。有关成本的更多信息，请参阅the section called “估算其他 CloudFront 指标的成本”。

必须为每个分配单独启用这些额外指标：

缓存命中率

由 CloudFront 从其缓存提供内容的所有可缓存请求的百分比。HTTP POST 和 PUT 请求及错误不视为可缓存请求。

来源延迟

对于从来源提供内容（而非从 CloudFront 缓存提供内容）的请求，从 CloudFront 接收请求，到开始向网络（而非查看器）提供响应为止所花费的总时间。这也称为首字节延迟或 time-to-first-byte。

按状态代码列出的错误率

响应的 HTTP 状态代码为 4xx 或 5xx 范围中的特定代码的所有查看器请求所占的百分比。此指标适用于以下所有错误代码：401、403、404、502、503 和 504。

启用其他指标

您可以在 CloudFront 控制台中、使用 AWS CloudFormation、AWS Command Line Interface (AWS CLI) 或 CloudFront API 启用其他指标。

Console

启用其他指标（控制台）

1. 登录 AWS Management Console，然后在 CloudFront 控制台中打开监控页面。

2. 选择要为其启用其他指标的分配，然后选择查看分配指标。
3. 选择 Manage additional metrics（管理其他指标）。
4. 在 Manage additional metrics（管理其他指标）窗口中，打开 Enabled（启用）。启用其他指标后，您可以关闭 Manage additional metrics（管理其他指标）窗口。

启用其他指标后，它们将显示在图表中。在每个图表上，总计值按 1 分钟的粒度显示。除了查看图表外，您还可以将指标报告下载为 CSV 文件。

您可以通过执行以下操作来自定义图表：

- 要更改图表中所显示信息的时间范围，请选择 1h（1 小时）、3h（3 小时）或其他范围，或指定自定义范围。
- 要更改 CloudFront 更新图表中信息的频率，请选择刷新图标旁边的下箭头，然后选择一个刷新间隔。默认刷新频率为 1 分钟，不过您可以选择 10 秒、2 分钟或其他选项。

要在 CloudWatch 控制台中查看 CloudFront 图形，请选择添加到控制面板。

AWS CloudFormation

要使用 AWS CloudFormation 打开其他指标，请使用 `AWS::CloudFront::MonitoringSubscription` 资源类型。以下示例以 YAML 格式显示 AWS CloudFormation 模板语法，用于启用其他指标。

```
Type: AWS::CloudFront::MonitoringSubscription
Properties:
  DistributionId: EDFDVBD6EXAMPLE
  MonitoringSubscription:
    RealtimeMetricsSubscriptionConfig:
      RealtimeMetricsSubscriptionStatus: Enabled
```

CLI

要使用 AWS Command Line Interface (AWS CLI) 管理其他指标，请使用下列命令之一：

为分配启用其他指标 (CLI)

- 使用 create-monitoring-subscription 命令，如以下示例所示。将 *EDFDVBD6EXAMPLE* 替换为要为其启用其他指标的分配的 ID。

```
aws cloudfront create-monitoring-subscription --
distribution-id EDFDVBD6EXAMPLE --monitoring-subscription
 RealtimeMetricsSubscriptionConfig={RealtimeMetricsSubscriptionStatus=Enabled}
```

查看是否为分配启用了其他指标 (CLI)

- 使用 get-monitoring-subscription 命令，如以下示例所示。将 *EDFDVBD6EXAMPLE* 替换为您正在检查的分配的 ID。

```
aws cloudfront get-monitoring-subscription --distribution-id EDFDVBD6EXAMPLE
```

为分配禁用其他指标 (CLI)

- 使用 delete-monitoring-subscription 命令，如以下示例所示。将 *EDFDVBD6EXAMPLE* 替换为要为其禁用其他指标的分配的 ID。

```
aws cloudfront delete-monitoring-subscription --distribution-id EDFDVBD6EXAMPLE
```

API

要使用 CloudFront API 管理其他指标，请使用下列 API 操作之一。

- 要为分配启用其他指标，请使用 CreateMonitoringSubscription。
- 要查看是否为分配启用了其他指标，请使用 GetMonitoringSubscription。
- 要为分配禁用其他指标，请使用 DeleteMonitoringSubscription。

有关这些 API 调用的更多信息，请参阅有关 AWS SDK 或其他 API 客户端的 API 参考文档。

估算其他 CloudFront 指标的成本

当您为某个分配启用其他指标时，CloudFront 会向美国东部（弗吉尼亚州北部）区域的 CloudWatch 发送最多 8 个指标。CloudWatch 为每个指标收取较低的固定费率。每个月对每个指标仅收取一次该费率（每个分配最多 8 个指标）。这是固定费率，因此无论 CloudFront 分配接收或发送的请求或响

应数量如何，您的成本都保持不变。有关每个指标的费率，请参阅 Amazon CloudWatch 定价页面和 CloudWatch 定价计算器。当您使用 CloudWatch API 检索指标时，将收取额外的 API 费用。

查看默认的 Lambda@Edge 函数指标

可以使用 CloudWatch 指标实时监控与 Lambda@Edge 函数相关的问题。使用这些指标无需额外付费。

当你将 Lambda@Edge 函数附加到 CloudFront 分配中的缓存行为时，Lambda 开始自动向 CloudWatch 发送指标。指标适用于所有 Lambda 区域，但要在 CloudWatch 控制台中查看指标或从 CloudWatch API 获取指标数据，您必须使用美国东部（弗吉尼亚州北部）区域 (`us-east-1`)。指标组名称的格式为：`AWS/CloudFront/`*`distribution-ID`*，其中 *`distribution-ID`* 是与 Lambda@Edge 函数关联的 CloudFront 分配的 ID。有关 Amazon CloudWatch 的更多信息，请参阅 Amazon CloudWatch 用户指南。

在 CloudFront 控制台的监控页面上，每个 Lambda@Edge 函数都会以图表形式显示以下这些默认指标：

- Lambda@Edge 的 `5xx` 错误率
- Lambda 执行错误
- Lambda 无效响应
- Lambda 节流

图表包含调用数、错误、限制等信息。在每个图表上，总计值按 1 分钟的粒度显示，按 AWS 区域分组。

如果您看到错误出现峰值需要调查，您可以选择一个函数，然后按 AWS 区域查看日志，直至您确定哪个函数在哪个 AWS 区域中导致了问题。有关排查 Lambda@Edge 错误的更多信息，请参阅：

- the section called “如何确定故障类型”
- AWS 上的内容分发的四个调试步骤

您可以通过执行以下操作来自定义图表：

- 要更改图表中所显示信息的时间范围，请选择 1h（1 小时）、3h（3 小时）或其他范围，或指定自定义范围。
- 要更改 CloudFront 更新图表中信息的频率，请选择刷新图标旁边的下箭头，然后选择一个刷新间隔。默认刷新频率为 1 分钟，不过您可以选择 10 秒、2 分钟或其他选项。

要在 CloudWatch 控制台中查看图表，请选择添加到控制面板。您必须使用美国东部（弗吉尼亚州北部）区域 (us-east-1) 在 CloudWatch 控制台中查看图表。

查看默认的 CloudFront Functions 指标

CloudFront Functions 将操作指标发送到 Amazon CloudWatch，以便您可以监控自己的函数。查看这些指标可帮助您解决、跟踪和调试问题。CloudFront Functions 向 CloudWatch 发布以下指标：

- 调用数 (`FunctionInvocations`) – 给定时间内开始（调用）函数的次数。
- 验证错误数 (`FunctionValidationErrors`) – 函数在给定时间段内产生的验证错误数。当函数成功运行但返回无效数据（无效的事件对象）时，会发生验证错误。
- 执行错误数 (`FunctionExecutionErrors`) – 给定时间内发生的执行错误数。当函数无法成功完成时，会发生执行错误。
- 计算利用率 (`FunctionComputeUtilization`) – 函数运行所花费的时间占最大允许时间的百分比。例如，值为 35 表示函数在最大允许时间的 35% 内完成。该指标是介于 0 到 100 之间的数字。

  如果此值达到或接近 100，表明该函数已用尽或接近用尽允许的执行时间，并且后续请求可能会受到限制。如果您的函数以 80% 或更高的利用率运行，建议您检查该函数以缩短执行时间并提高利用率。例如，您可能只想记录错误，那么请简化任何复杂的正则表达式或取消对复杂 JSON 对象的不必要解析。
- 限制 (`FunctionThrottles`) – 在给定时间段内函数受到限制的次数。下列原因可能导致函数受到限制：
  - 该函数持续超过执行所允许的最长时间
  - 该函数导致编译错误
  - 每秒的请求数异常高

CloudFront KeyValueStore 还向 Amazon CloudWatch 发送以下运营指标：

- 读取请求（`KvsReadRequests`）– 该函数在给定时段内从键值存储成功读取的次数。
- 读取错误（`KvsReadErrors`）– 该函数在给定时段内从键值存储读取失败的次数。

要在 CloudFront 控制台中查看这些指标，请转至监控页面。要查看特定函数的图形，请选择 Functions（函数），选择函数，然后选择 View function metrics（查看函数指标）。

所有这些指标都在 CloudFront 命名空间中发布到美国东部（弗吉尼亚州北部）区域 (us-east-1) 的 CloudWatch。您还可以在 CloudWatch 控制台中查看这些指标。在 CloudWatch 控制台中，您可以查看每个函数或每个分配中每个函数的指标。

您还可以使用 CloudWatch 根据这些指标设置警报。例如，您可以根据执行时间 (FunctionComputeUilization) 指标设置警报，该指标表示函数运行所花费的可用时间百分比。当执行时间在一定时间内达到特定值时（例如，连续 15 分钟内超过 70% 的可用时间）时，警报将触发。您可以在创建警报时指定警报的值及其时间单位。

> **Note**
>
> CloudFront Functions 仅为处理生产请求和响应而运行的 LIVE 阶段中的函数向 CloudWatch 发送指标。测试函数时，CloudFront 不会向 CloudWatch 发送任何指标。测试输出包含有关错误、计算利用率和函数日志（console.log() 语句）的信息，但这些信息不会发送到 CloudWatch。

有关如何使用 CloudWatch API 获取这些指标的信息，请参阅the section called “使用 API 获取指标”。

为指标创建警报

在 CloudFront 控制台中，您可以根据特定 CloudFront 指标设置警报，以接收 Amazon Simple Notification Service (Amazon SNS) 的通知。您可以在 CloudFront 控制台中的警报页面上设置警报。

要在控制台中创建警报，请指定以下值：

指标

您正在为其创建警报的指标。

分配

您正在为其创建警报的 CloudFront 分配。

警报名称

警报的名称。

Send a notification to (发送通知到)

当此指标触发警报时，要将通知发送到的 Amazon SNS 主题。

Whenever ***<metric> <operator> <value>*** (每当 <指标> <运算符> <值>)

指定 CloudWatch 应何时触发警报并向 Amazon SNS 主题发送通知。例如，要在 5xx 错误率超过 1% 时接收通知，请指定以下内容：

Whenever Average of 5xxErrorRate (每当 5xxErrorRate 的平均值)**> 1**

请注意以下有关指定值的事项：

- 只能输入不带标点符号的整数。例如，要指定一千，请输入 **1000**。
- 对于 4xx、5xx 和总错误率，您指定的值为百分比。
- 对于请求、已下载的字节和已上传的字节，指定的值带有单位。例如，1073742000 个字节。

For at least ***<number>*** consecutive periods of ***<time period>*** (对于 <时间段> 的至少 <数字> 个连续期间)

指定在所选持续时间的多少个连续时间段内指标满足标准，CloudWatch 才会触发警报。当您选择一个值时，目标是在一个值之间实现适当的平衡，该值不会因临时或短暂的问题而发出警报，但对持续或实际问题确实会发出警报。

以 CSV 格式下载指标数据

您可以下载 CSV 格式的 CloudFront 分配的 CloudWatch 指标数据。在 CloudFront 控制台中，针对特定分配查看分配指标时，可以下载数据。

该报告的相关信息

该报告的前几行包含以下信息：

Version

CloudFront 报告版本。

报告

报告的名称。

DistributionID

您运行其报告的分配的 ID。

StartDateUTC

您运行报告的日期范围的开始日期，采用协调世界时 (UTC)。

EndDateUTC

您运行报告的日期范围的结束日期，采用协调世界时 (UTC)。

GeneratedTimeUTC

您运行报告的日期和时间，采用协调世界时 (UTC)。

粒度

报告中每一行的时间段，例如 ONE_MINUTE。

指标报告中的数据

该报告包括以下值：

DistributionID

您运行其报告的分配的 ID。

FriendlyName

分配的备用域名 (CNAME)（如果有）。如果分配没有备用域名，则该列表会包括该分配的原始域名。

TimeBucket

数据适用的小时或天，采用协调世界时 (UTC)。

请求

特定时间段内对所有 HTTP 状态代码（例如，200、404 等）及所有方法（例如，GET、HEAD、POST 等）的请求总数。

BytesDownloaded

查看器在特定时间段内为指定的分配下载的字节数。

BytesUploaded

查看器在特定时间段内为指定的分配上传的字节数。

TotalErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 4xx 或 5xx 错误的请求百分比。

4xxErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 4xx 错误的请求百分比。

5xxErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 5xx 错误的请求百分比。

如果您已为分配启用了其他指标，则报告还包含以下附加值：

401ErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 401 错误的请求百分比。

403ErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 403 错误的请求百分比。

404ErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 404 错误的请求百分比。

502ErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 502 错误的请求百分比。

503ErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 503 错误的请求百分比。

504ErrorRatePct

对于特定时间段内的指定分配，HTTP 状态代码为 504 错误的请求百分比。

OriginLatency

对于从来源提供内容（而非从 CloudFront 缓存提供内容）的请求，从 CloudFront 接收请求，到开始向网络（而非查看器）提供响应为止所花费的总时间（以毫秒为单位）。这也称为首字节延迟或 time-to-first-byte。

CacheHitRate

由 CloudFront 从其缓存提供内容的所有可缓存请求的百分比。HTTP POST 和 PUT 请求及错误不视为可缓存请求。

使用 CloudWatch API 获取指标

您可以使用 Amazon CloudWatch API 或 CLI 在您构建的程序或应用程序中获取 CloudFront 指标。您可以使用原始数据构建自己的自定义控制面板、自己的警报工具等。

要从 CloudWatch API 获取 CloudFront 指标，您必须使用美国东部（弗吉尼亚州北部）区域 (us-east-1)。您还需要了解每个指标的特定值和类型。

主题

所有 CloudFront 指标的值

以下值适用于所有 CloudFront 指标：

`Namespace`

`Namespace` 的值始终为 `AWS/CloudFront`。

维度

每个 CloudFront 指标都具有以下两个维度：

`DistributionId`

要获取其指标的 CloudFront 分配的 ID。

`FunctionName`

要获取指标的函数的名称（在 CloudFront Functions 中）。

此维度仅适用于函数。

`Region`

`Region` 的值始终是 `Global`，因为 CloudFront 是全球性服务。

> **Note**
>
> 要从 CloudWatch API 获取 CloudFront 指标，您必须使用美国东部（弗吉尼亚州北部）区域 (us-east-1)。

CloudFront 分配指标的值

使用以下列表中的信息从 CloudWatch API 获取有关特定 CloudFront 分配指标的详细信息。仅当您为分配启用了其他指标时，其中某些指标才可用。

> **Note**
>
> 每个指标只有一个适用的统计数据 `Average` 或 `Sum`。以下列表指定了适用于该指标的统计数据。

4xx 错误率

响应的 HTTP 状态代码为 `4xx` 的所有查看器请求所占的百分比。

- 指标名称：`4xxErrorRate`
- Valid statistic (有效统计数据)：`Average`
- 单位：`Percent`

401 错误率

响应的 HTTP 状态代码为 `401` 的所有查看器请求所占的百分比。要获取此指标，您必须首先启用其他指标。

- 指标名称：`401ErrorRate`
- Valid statistic (有效统计数据)：`Average`
- 单位：`Percent`

403 错误率

响应的 HTTP 状态代码为 `403` 的所有查看器请求所占的百分比。要获取此指标，您必须首先启用其他指标。

- 指标名称：`403ErrorRate`
- Valid statistic (有效统计数据)：`Average`
- 单位：`Percent`

404 错误率

响应的 HTTP 状态代码为 `404` 的所有查看器请求所占的百分比。要获取此指标，您必须首先启用其他指标。

- 指标名称：`404ErrorRate`

- Valid statistic (有效统计数据)：Average
- 单位：Percent

5xx 错误率

响应的 HTTP 状态代码为 5xx 的所有查看器请求所占的百分比。

- 指标名称：5xxErrorRate
- Valid statistic (有效统计数据)：Average
- 单位：Percent

502 错误率

响应的 HTTP 状态代码为 502 的所有查看器请求所占的百分比。要获取此指标，您必须首先启用其他指标。

- 指标名称：502ErrorRate
- Valid statistic (有效统计数据)：Average
- 单位：Percent

503 错误率

响应的 HTTP 状态代码为 503 的所有查看器请求所占的百分比。要获取此指标，您必须首先启用其他指标。

- 指标名称：503ErrorRate
- Valid statistic (有效统计数据)：Average
- 单位：Percent

504 错误率

响应的 HTTP 状态代码为 504 的所有查看器请求所占的百分比。要获取此指标，您必须首先启用其他指标。

- 指标名称：504ErrorRate
- Valid statistic (有效统计数据)：Average
- 单位：Percent

已下载字节

查看器针对 GET、HEAD 和 OPTIONS 请求下载的字节总数。

- Metric name (指标名称)：BytesDownloaded
- Valid statistic (有效统计数据)：Sum

- 单位：None

已上传字节

查看器通过 CloudFront 使用 POST 和 PUT 请求上传到您的源的字节总数。

- Metric name (指标名称)：BytesUploaded
- Valid statistic (有效统计数据)：Sum
- 单位：None

缓存命中率

由 CloudFront 从其缓存提供内容的所有可缓存请求的百分比。HTTP POST 和 PUT 请求及错误不视为可缓存请求。要获取此指标，您必须首先启用其他指标。

- 指标名称：CacheHitRate
- Valid statistic (有效统计数据)：Average
- 单位：Percent

来源延迟

对于从来源提供内容（而非从 CloudFront 缓存提供内容）的请求，从 CloudFront 接收请求，到开始向网络（而非查看器）提供响应为止所花费的总时间（以毫秒为单位）。这也称为首字节延迟或 time-to-first-byte。要获取此指标，您必须首先启用其他指标。

- 指标名称：OriginLatency
- Valid statistic (有效统计数据)：Percentile
- 单位：Milliseconds

> **Note**
>
> 要从 CloudWatch API 获取 Percentile 统计信息，请使用 ExtendedStatistics 参数，而不是 Statistics。有关更多信息，请参阅 Amazon CloudWatch API 参考中的 GetMetricStatistics 或 AWS 开发工具包的参考文档。

请求

针对所有 HTTP 方法以及 HTTP 和 HTTPS 请求，CloudFront 收到的查看器请求总数。

- Metric name (指标名称)：Requests
- Valid statistic (有效统计数据)：Sum
- 单位：None

总错误率

响应的 HTTP 状态代码为 4xx 或 5xx 的所有查看器请求所占的百分比。

- 指标名称：TotalErrorRate
- Valid statistic (有效统计数据)：Average
- 单位：Percent

CloudFront 函数指标的值

使用以下列表中的信息从 CloudWatch API 获取有关特定 CloudFront 函数指标的详细信息。

> **Note**
>
> 每个指标只有一个适用的统计数据 Average 或 Sum。以下列表指定了适用于该指标的统计数据。

调用

给定时间内开始（调用）函数的次数。

- Metric name (指标名称)：FunctionInvocations
- Valid statistic (有效统计数据)：Sum
- 单位：None

验证错误

函数在给定时间段内产生的验证错误数。当函数成功运行但返回无效数据（无效的事件对象）时，会发生验证错误。

- Metric name (指标名称)：FunctionValidationErrors
- Valid statistic (有效统计数据)：Sum
- 单位：None

执行错误

给定时间内发生的执行错误数。当函数无法成功完成时，会发生执行错误。

- Metric name (指标名称)：FunctionExecutionErrors
- Valid statistic (有效统计数据)：Sum
- 单位：None

计算利用率

函数运行所花费的时间占最大允许时间的百分比 (0-100)。例如，值为 35 表示函数在最大允许时间的 35% 内完成。

- 指标名称：FunctionComputeUtilization
- Valid statistic (有效统计数据)：Average
- 单位：Percent

节流

在给定时间段内函数受到限制的次数。

- 指标名称：FunctionThrottles
- Valid statistic (有效统计数据)：Sum
- 单位：None

CloudFront 和边缘函数日志记录

Amazon CloudFront 提供不同类型的日志记录。您可以记录到达您的 CloudFront 分配的查看器请求，也可以在 AWS 账户中记录 CloudFront 服务活动（API 活动）。还可以从您的边缘计算函数中获取日志。

记录请求

CloudFront 提供了以下方法来记录到达分配的请求。

标准日志（访问日志）

CloudFront 标准日志提供有关向分配发出的每个请求的详细记录。在许多情况下（包括安全和访问审计），这些日志很有用。

CloudFront 标准日志将传送到您选择的 Amazon S3 存储桶。针对标准日志，CloudFront 不收取费用，但存储和访问日志文件将产生 Amazon S3 费用。

有关更多信息，请参阅 使用标准日志（访问日志）。

实时日志

CloudFront 实时日志可以实时提供有关向分配发出的请求的信息（日志记录在收到请求后的几秒钟内传输）。您可以选择实时日志的采样率 – 即希望接收实时日志记录的请求的百分比。您还可以选择希望在日志记录中接收的特定字段。

CloudFront 实时日志将传送到 Amazon Kinesis Data Streams 中您选择的数据流。CloudFront 除了收取因使用 Kinesis Data Streams 产生的费用外，还针对实时日志进行收费。

有关更多信息，请参阅 实时日志。

记录边缘函数

可以使用 Amazon CloudWatch Logs 获取边缘函数（包括 Lambda@Edge 和 CloudFront Functions）的日志。可以使用 CloudWatch 控制台或 CloudWatch Logs API 访问日志。有关更多信息，请参阅 the section called “边缘函数日志”。

记录服务活动

您可以使用 AWS CloudTrail 在您的 AWS 账户中记录 CloudFront 服务活动（API 活动）。CloudTrail 提供了用户、角色或 AWS 服务在 CloudFront 中所执行 API 操作的记录。使用 CloudTrail 收集的信息，您可以确定向 CloudFront 发出了什么 API 请求、发出请求的 IP 地址、何人发出的请求、发出请求的时间以及其他详细信息。

有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon CloudFront API 调用。

主题

- 配置和使用标准日志（访问日志）
- 实时日志
- 边缘函数日志
- 使用 AWS CloudTrail 记录 Amazon CloudFront API 调用

配置和使用标准日志（访问日志）

您可以将 CloudFront 配置为创建包含有关 CloudFront 接收的每个用户请求的详细信息的日志文件。这些日志称为标准日志，也称为访问日志。如果启用标准记录，您还可以指定希望 CloudFront 在其中保存文件的 Amazon S3 存储桶。

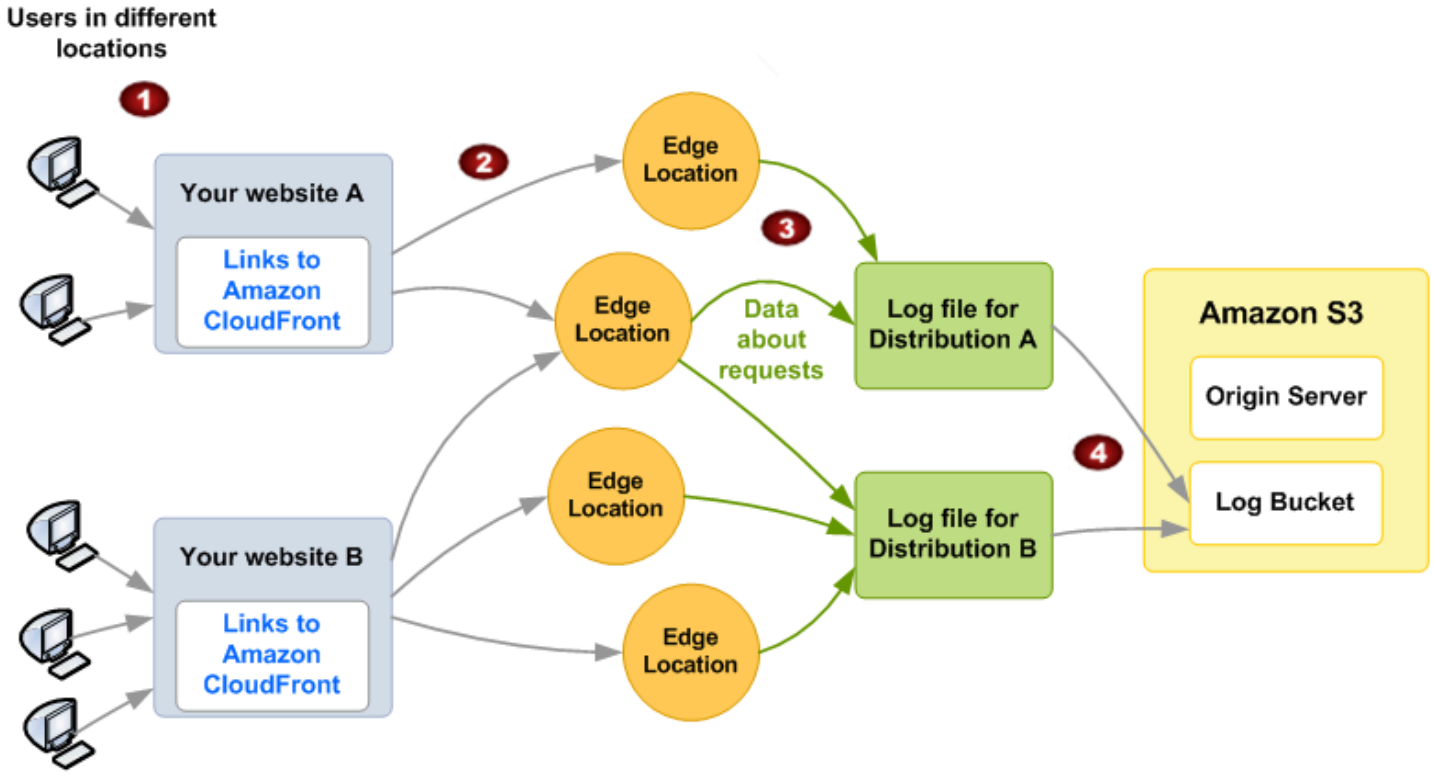
您可以在创建或更新分配时启用标准日志。有关更多信息，请参阅分配设置参考。

CloudFront 还提供了实时日志，可以实时向您提供有关向分配发出的请求的信息（日志在收到请求后的几秒钟内传输）。您可以使用实时日志来进行监控和分析，并根据内容交付性能采取相应措施。有关更多信息，请参阅实时日志。

主题

标准日志记录的工作方式

下图显示了 CloudFront 如何记录有关对您的对象的请求的信息。



下面介绍 CloudFront 如何记录有关对您的对象的请求的信息（如上图中所示）。

1. 在此图中，您有 A 和 B 两个网站和两个对应的 CloudFront 分配。用户使用与您的分配相关联的 URL 来请求您的对象。
2. CloudFront 会将每个请求路由到适当的边缘站点。
3. CloudFront 将每个请求的数据写入分配特定的日志文件。在本示例中，与分配 A 有关的请求信息将写入分配 A 的日志文件，与分配 B 有关的请求信息将写入分配 B 的日志文件。
4. CloudFront 会定期将有关分配的日志文件保存在您启用日志记录时指定的 Amazon S3 存储桶。然后，CloudFront 开始将有关后续请求的信息保存在关于该分配的新日志文件中。

如果在给定时间内无用户访问您的内容，您在该时间内不会接收任何日志文件。

日志文件中的每个条目分别提供有关单个请求的详细信息。有关文件格式的更多信息，请参阅标准日志文件格式。

> **Note**
>
> 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。CloudFront 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。当访问日志中省略了日志条目时，访问日志中的条目数将与 AWS 账单和使用率报告中出现的使用率不匹配。

为您的标准日志选择 Amazon S3 存储桶

当您对分配启用日志记录时，需要指定您希望 CloudFront 将日志文件存储到其中的 Amazon S3 存储桶。如果您使用 Amazon S3 作为源，建议您不要对日志文件使用同一个桶；使用单独的桶可简化维护。

> **Important**
>
> 不要选择 S3 对象所有权设置为强制桶拥有者的 Amazon S3 存储桶。该设置对存储桶以及其中的对象禁用 ACL，这会阻止 CloudFront 将日志文件传输到存储桶。

Important

请勿在以下任何区域中选择 Amazon S3 存储桶，因为 CloudFront 不会将标准日志传送到这些区域中的存储桶：

- 非洲（开普敦）
- 亚太地区（香港）
- 亚太地区（海得拉巴）
- 亚太地区（雅加达）
- 亚太地区（墨尔本）
- 加拿大西部（卡尔加里）
- 欧洲地区（米兰）
- 欧洲（西班牙）
- 欧洲（苏黎世）
- 以色列（特拉维夫）
- 中东（巴林）
- 中东（阿联酋）

可将多个分配的日志文件存储在同一存储桶中。当您启用日志记录时，可为文件名指定一个可选前缀，以便您可以跟踪哪些日志文件与哪些分配相关联。

配置标准日志记录和访问您的日志文件所需的权限

Important

从 2023 年 4 月开始，您将需要为用于 CloudFront 标准日志的新 S3 存储桶启用 S3 访问控制列表 (ACL)。ACL 可以在存储桶创建步骤中启用，也可以在创建存储桶之后启用。
有关这些更改的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的新 S3 桶的原定设置常见问题解答和《AWS 新闻博客》中的提醒：Amazon S3 安全更改将于 2023 年 4 月发布。

您的 AWS 账户必须对您为日志文件指定的存储桶拥有以下权限：

- 存储桶的 S3 访问控制列表 (ACL) 必须向您授予 FULL_CONTROL。如果您是存储桶拥有者，则默认情况下，您的账户具有此权限。如果您不是，则存储桶拥有者必须更新存储桶的 ACL。
- `s3:GetBucketAcl`
- `s3:PutBucketAcl`

请注意以下几点：

存储桶的 ACL

当您创建或更新分配并启用日志记录时，CloudFront 会使用这些权限更新存储桶的 ACL，以授予 `awslogsdelivery` 账户 FULL_CONTROL 权限。`awslogsdelivery` 账户将日志文件写入存储桶。如果您的账户没有更新 ACL 所需的权限，则创建或更新分配将会失败。

在某些情况下，如果以编程方式提交请求以创建存储桶，但具有指定名称的存储桶已存在，则 S3 将存储桶上的权限重置为默认值。如果您已将 CloudFront 配置为将访问日志保存在 S3 存储桶中，并停止获取该存储桶中的日志，请检查存储桶上的权限，以确保 CloudFront 具有必要的权限。

恢复存储桶的 ACL

如果您删除对 `awslogsdelivery` 账户的权限，则 CloudFront 无法将日志保存到 S3 存储桶。要使 CloudFront 能够再次开始保存您的分配的日志，请通过执行以下操作之一恢复 ACL 权限：

- 在 CloudFront 中对您的分配禁用日志记录，然后再次启用它。有关更多信息，请参阅分配设置参考。
- 通过在 Amazon S3 控制台中导航到 S3 存储桶并添加权限，手动为 `awslogsdelivery` 添加 ACL 权限。要为 `awslogsdelivery` 添加 ACL，您必须提供账户的规范 ID，如下所示：

  `c4c1ede66af53448b93c283ce9448c4ba468c9432aa01d700d3878632f77d2d0`

  有关将 ACL 添加到 S3 存储桶的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的如何设置 ACL 存储桶权限？。

每个日志文件的 ACL

除了存储桶上的 ACL 之外，每个日志文件上还有一个 ACL。存储桶拥有者对每个日志文件均具有 FULL_CONTROL 权限，分配所有者（如果与存储桶拥有者不同）没有权限，而 `awslogsdelivery` 账户具有读取和写入权限。

禁用日志记录

如果您禁用日志记录，则 CloudFront 并不会删除存储桶或日志文件的 ACL。如果您愿意，可以自己撤销。

SSE-KMS 存储桶的必需密钥策略

如果您的标准日志的 S3 存储桶通过 AWS KMS keys (SSE-KMS) 的客户托管密钥使用服务器端加密，您必须为您的客户托管密钥的密钥策略添加以下语句。这样一来，CloudFront 可以将日志文件写入存储桶中。（您无法将 SSE-KMS 与 AWS 托管式密钥结合使用，因为 CloudFront 无法将日志文件写入桶。）

```
{
    "Sid": "Allow CloudFront to use the key to deliver logs",
    "Effect": "Allow",
    "Principal": {
        "Service": "delivery.logs.amazonaws.com"
    },
    "Action": "kms:GenerateDataKey*",
    "Resource": "*"
}
```

如果标准日志的 S3 存储桶将 SSE-KMS 与 S3 存储桶密钥结合使用，您还需要将 `kms:Decrypt` 权限添加到策略语句中。在这种情况下，完整的策略语句如下所示。

```
{
    "Sid": "Allow CloudFront to use the key to deliver logs",
    "Effect": "Allow",
    "Principal": {
        "Service": "delivery.logs.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey*",
        "kms:Decrypt"
    ],
    "Resource": "*"
}
```

文件名格式

CloudFront 保存在您的 Amazon S3 存储桶中的每个日志文件的名称使用以下文件名格式：

<optional prefix>/*<distribution ID>*.*YYYY*-*MM*-*DD*-*HH*.*unique-ID*.gz

日期和时间用协调世界时 (UTC) 表示。

例如，如果您使用 example-prefix 作为前缀，并且您的分配 ID 为 EMLARXS9EXAMPLE，则您的文件名看起来类似于以下内容：

example-prefix/EMLARXS9EXAMPLE.2019-11-14-20.RT4KCN4SGK9.gz

当您对分配启用日志记录时，可为文件名指定一个可选前缀，以便您可以跟踪哪些日志文件与哪些分配相关联。如果您包含日志文件前缀的值，并且您的前缀不以正斜杠 (/) 结尾，CloudFront 则会自动追加一个。如果您的前缀以正斜杠结尾，则 CloudFront 不会添加另一个斜杠。

文件名末尾的 .gz 表示 CloudFront 已使用 gzip 压缩日志文件。

标准日志文件传输计时

CloudFront 一个小时内会为分配提交若干次标准日志。一般而言，日志文件包含有关 CloudFront 在给定时间段内收到的请求的信息。CloudFront 通常会在日志中所显示事件发生后的一个小时内将该时间段内的日志文件传输至 Amazon S3 存储桶。但是，请注意，某个时间段内的某些或所有日志文件条目有时可延迟长达 24 小时。当日志条目延迟后，CloudFront 会将它们保存在其文件名称包括发生请求的时间段的日期和时间（而不是文件传输的日期和时间）的日志文件中。

创建日志文件时，CloudFront 将在日志文件涵盖的时间段内从收到对象请求的所有边缘站点整合分配信息。

CloudFront 可为某个时间段保存多个文件，具体取决于 CloudFront 收到的针对与分配相关联的对象的请求数。

CloudFront 在您启用日志记录后大约四个小时开始可靠地传输访问日志。您可能会获得一些在此时间之前的访问日志。

> **Note**
>
> 如果在此期间没有用户请求您的对象，您就不会收到该期间的任何日志文件。

CloudFront 还提供了实时日志，可以实时向您提供有关向分配发出的请求的信息（日志在收到请求后的几秒钟内传输）。您可以使用实时日志来进行监控和分析，并根据内容交付性能采取相应措施。有关更多信息，请参阅实时日志。

当请求 URL 或标头超出最大大小时如何记录请求

如果所有请求标头（包括 Cookie）的总大小超过 20KB，或者如果 URL 超过 8192 字节，则 CloudFront 无法完整解析并记录请求。由于未记录该请求，因此您在日志文件中看不到返回的 HTTP 错误状态代码。

如果请求正文超出最大大小，则会记录请求，包括 HTTP 错误状态代码。

分析标准日志

由于您每小时可以收到多个访问日志，因此建议您将给定时段内接收的所有日志文件合并成一个文件。然后，您可更准确更全面地分析该时期内的数据。

分析访问日志的一种方式是使用 Amazon Athena。Athena 是一项交互式查询服务，可以帮助您为 AWS 服务（包括 CloudFront）分析数据。要了解更多信息，请参阅《Amazon Athena 用户指南》中的查询 Amazon CloudFront 日志 。

此外，以下 AWS 博客文章讨论了分析访问日志的一些方式。

- Amazon CloudFront 请求日志记录（针对通过 HTTP 提供的内容）
- 增强的 CloudFront 日志，现在包含查询字符串

> **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。CloudFront 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。当访问日志中省略了日志条目时，访问日志中的条目数将与 AWS 使用率和账单报告中出现的使用率不匹配。

编辑标准日志记录设置

您可通过使用 CloudFront 控制台或 CloudFront API 来启用或禁用日志记录、更改存储日志的 Amazon S3 存储桶以及更改日志文件的前缀。您对日志记录设置的更改将在 12 小时内生效。

有关更多信息，请参阅以下主题：

- 要使用 CloudFront 控制台更新分配，请参阅更新分配。
- 要使用 CloudFront API 更新分配，请参阅 Amazon CloudFront API 参考中的 UpdateDistribution。

从 Amazon S3 存储桶中删除标准日志文件

CloudFront 不会自动从您的 Amazon S3 存储桶中删除日志文件。有关删除 Amazon S3 存储桶中日志文件的信息，请参阅以下主题：

- 使用 Amazon S3 控制台：《Amazon Simple Storage Service 控制台用户指南》中的删除对象。
- 使用 REST API：《Amazon Simple Storage Service API 参考》中的 DeleteObject。

标准日志文件格式

日志文件中的每个条目分别提供有关单个查看器请求的详细信息。日志文件具有以下特征：

- 使用 W3C 扩展日志文件格式。
- 包含制表符分隔的值。
- 包含不一定按时间顺序排列的记录。
- 包含两个标题行：一个具有文件格式版本，另一个列出了包含在每个记录中的 W3C 字段。
- 包含字段值中的空格和某些其他字符的 URL 编码等效值。

  URL 编码的等效值用于以下字符：

  - ASCII 字符代码 0 到 32（含这两个值）
  - ASCII 字符代码 127 及更大值
  - 下表中的所有字符

  URL 编码标准在 RFC 1738 中定义。

URL 编码值	字符
%3C	<
%3E	>
%22	"
%23	#
%25	%

URL 编码值	字符
%7B	{
%7D	}
%7C	|
%5C	\
%5E	^
%7E	~
%5B	[
%5D	]
%60	`
%27	'
%20	空格

标准日志文件字段

分配的日志文件包含 33 个字段。以下列表包含每个字段名称（按顺序）以及该字段中信息的描述。

1. **date**

   事件发生日期的格式为 YYYY-MM-DD。例如，2019-06-30。日期和时间用协调世界时 (UTC) 表示。对于 WebSocket 连接，这是关闭连接的日期。

2. **time**

   CloudFront 服务器完成响应请求的时间（采用 UTC 时间），例如 01:42:39。对于 WebSocket 连接，这是关闭连接的时间。

3. **x-edge-location**

服务请求的边缘站点。每个边缘站点由三个字母的代码和任意分配的数字来确定（例如，DFW3）。三个字母代码通常对应邻近边缘站点的地理位置的机场的国际航空协会（IATA）机场代码。(这些缩写将来可能会更改。)

4. **sc-bytes**

   服务器在响应请求时向查看器提供的字节的总数，包括标头。对于 WebSocket 连接，这是通过连接从服务器发送到客户端的字节的总数。

5. **c-ip**

   已发出请求的查看器的 IP 地址，例如 `192.0.2.183` 或 `2001:0db8:85a3::8a2e:0370:7334`。如果查看器已使用 HTTP 代理或负载均衡器发送请求，则此字段的值为该代理或负载均衡器的 IP 地址。另请参阅 `x-forwarded-for` 字段。

6. **cs-method**

   从查看器接收的 HTTP 请求方法。

7. **cs(Host)**

   CloudFront 分配的域名（例如，d111111abcdef8.cloudfront.net）。

8. **cs-uri-stem**

   请求 URL 中标识路径和对象的部分（例如 `/images/cat.jpg`）。URL 和查询字符串中的问号 (?) 不包含在日志中。

9. **sc-status**

   包含下列值之一：

   - 服务器响应的 HTTP 状态代码（例如 `200`）。
   - `000`，表示查看器已在服务器可以响应请求之前关闭连接。如果查看器在服务器开始发送响应后关闭连接，则此字段包含服务器开始发送的响应的 HTTP 状态代码。

10. **cs(Referer)**

    请求中的 `Referer` 标头的值。这是发出请求的域的名称。常见引用站点包括搜索引擎、直接链接到您的对象的其他网站及您自己的网站。

11. **cs(User-Agent)**

    请求中的 `User-Agent` 标头的值。`User-Agent` 标头标识请求的来源，例如提交请求的设备和浏览器的类型，如果请求来自搜索引擎，则标识具体的搜索引擎。

12.**cs-uri-query**

请求 URL 的查询字符串部分（如果有）。

如果 URL 不包含查询字符串，则此字段的值为连字符 (-)。有关更多信息，请参阅根据查询字符串参数缓存内容。

13.**cs(Cookie)**

请求中的 Cookie 标头，包括名称/值对和关联的属性。

如果您启用了 Cookie 日志记录，无论您选择将哪种 Cookie 转发到源，CloudFront 都将记录所有请求中的 Cookie。当请求不包含 Cookie 标头时，此字段的值为连字符 (-)。有关 cookies 的更多信息，请参阅 根据 Cookie 缓存内容。

14.**x-edge-result-type**

在最后一个字节离开服务器后服务器如何对响应进行分类。在某些情况下，结果类型可能会在服务器准备发送响应的时间与完成发送响应的时间之间发生变化。另请参阅 x-edge-response-result-type 字段。

例如，在 HTTP 流中，假设服务器在缓存中发现流的一个区段。在这种情况下，此字段的值通常为 Hit。但是，如果查看器在服务器传送整个区段之前关闭连接，则最终结果类型（以及此字段的值）为 Error。

WebSocket 连接将具有此字段的值 Miss，因为内容不可缓存，并直接通过代理回到源。

可能的值包括：

- Hit – 服务器从缓存中将对象提供给查看器。
- RefreshHit – 服务器在缓存中找到了对象，但该对象已过期，因此，服务器联系了源，以验证缓存是否具有该对象的最新版本。
- Miss – 缓存中的对象无法满足请求，因此，服务器将请求转发到源并将结果返回到查看器。
- LimitExceeded – 请求被拒绝，因为超出了 CloudFront 配额（以前称为限制）。
- CapacityExceeded – 服务器返回了 HTTP 503 状态代码，因为它在请求时没有足够的容量来服务对象。
- Error – 通常，这意味着请求会导致客户端错误（sc-status 字段的值在 4xx 范围内）或服务器错误（sc-status 字段的值在 5xx 范围内）。如果 sc-status 字段的值为 200，或者如果此字段的值为 Error 且 x-edge-response-result-type 字段的值不是 Error，则表示 HTTP 请求已成功，但客户端在接收所有字节之前断开连接。

- `Redirect` – 服务器已根据分发设置将查看器从 HTTP 重定向到 HTTPS。

15 **`x-edge-request-id`**

唯一地标识请求的不透明字符串。CloudFront 还会在 `x-amz-cf-id` 响应标头中发送此字符串。

16 **`x-host-header`**

查看器在该请求的 `Host` 标头中包含的值。如果您在对象 URL 中使用 CloudFront 域名（如 d111111abcdef8.cloudfront.net），则此字段将包含该域名。如果在您的对象 URL（例如 www.example.com）中使用的是备用域名（CNAME），则此字段将包含备用域名。

如果您使用的是备用域名，请参阅与您的分配关联的域名的字段 7 中的 `cs(Host)`。

17 **`cs-protocol`**

查看器请求的协议（`http`、`https`、`ws` 或 `wss`）。

18 **`cs-bytes`**

查看器包含在请求中的数据字节总数，包括标头。对于 WebSocket 连接，这是通过连接从客户端发送到服务器的字节的总数。

19 **`time-taken`**

服务器收到查看器的请求的时间与服务器将响应的最后一个字节写入输出队列的时间之间相隔的秒数（精确至千分之一秒，例如 0.082），以服务器上测量的时间为准。从查看器的角度看，由于网络延迟和 TCP 缓冲的原因，获得完整响应的总时间将会超过该值。

20 **`x-forwarded-for`**

如果查看器已使用 HTTP 代理或负载均衡器发送请求，则 `c-ip` 字段的值为该代理或负载均衡器的 IP 地址。在本例中，此字段为发出请求的查看器的 IP 地址。此字段可以包含多个逗号分隔的 IP 地址。每个 IP 地址可以是 IPv4 地址（如 `192.0.2.183`）或 IPv6 地址（如 `2001:0db8:85a3::8a2e:0370:7334`）。

如果查看器未使用 HTTP 代理或负载均衡器，则此字段的值为连字符 (-)。

21 **`ssl-protocol`**

如果请求使用了 HTTPS，则此字段包含查看器和服务器协商用于传输请求和响应的 SSL/TLS 协议。有关可能的值的列表，请参阅查看器和 CloudFront 之间支持的协议和密码中支持的 SSL/TLS 协议。

如果字段 17 中的 `cs-protocol` 为 `http`，则此字段的值为连字符 (-)。

22 **ssl-cipher**

如果请求使用了 HTTPS，则此字段包含查看器和服务器协商用于加密请求和响应的 SSL/TLS 密码。有关可能的值的列表，请参阅查看器和 CloudFront 之间支持的协议和密码中支持的 SSL/TLS 密码。

如果字段 17 中的 cs-protocol 为 http，则此字段的值为连字符 (-)。

23 **x-edge-response-result-type**

服务器在将响应返回到查看器之前如何对响应进行分类。另请参阅 x-edge-result-type 字段。可能的值包括：

- Hit – 服务器从缓存中将对象提供给查看器。
- RefreshHit – 服务器在缓存中找到了对象，但该对象已过期，因此，服务器联系了源，以验证缓存是否具有该对象的最新版本。
- Miss – 缓存中的对象无法满足请求，因此，服务器将请求转发到源服务器并将结果返回到查看器。
- LimitExceeded – 请求被拒绝，因为超出了 CloudFront 配额（以前称为限制）。
- CapacityExceeded – 服务器返回了 503 错误，因为它在请求时没有足够的容量来服务对象。
- Error – 通常，这意味着请求会导致客户端错误（sc-status 字段的值在 4xx 范围内）或服务器错误（sc-status 字段的值在 5xx 范围内）。

  如果 x-edge-result-type 字段的值为 Error，而此字段的值不是 Error，则客户端在下载完成前已断开连接。
- Redirect – 服务器已根据分发设置将查看器从 HTTP 重定向到 HTTPS。

24 **cs-protocol-version**

查看器在请求中指定的 HTTP 版本。可能的值包括 HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2.0 和 HTTP/3.0。

25 **fle-status**

在为分配配置字段级加密时，此字段包含一个指示是否已成功处理请求正文的代码。如果服务器成功处理了请求正文，加密了指定字段中的值并将请求转发到源，则此字段的值为 Processed。在这种情况下，x-edge-result-type 的值仍可以指示客户端或服务器端错误。

此字段的可能值包括：

- ForwardedByContentType – 由于没有配置内容类型，因此服务器将请求转发到了源，而不进行解析或加密。
- ForwardedByQueryArgs – 由于请求包含的查询参数不在字段级加密的配置中，因此服务器将请求转发到了源，而不进行解析或加密。
- ForwardedDueToNoProfile – 由于在字段级加密的配置中没有指定配置文件，因此服务器将请求转发到了源，而不进行解析或加密。
- MalformedContentTypeClientError – 由于 Content-Type 标头值的格式无效，因此，服务器拒绝了该请求并向查看器返回了 HTTP 400 状态代码。
- MalformedInputClientError – 由于请求正文的格式无效，因此服务器拒绝了该请求，并向查看器返回了 HTTP 400 状态代码。
- MalformedQueryArgsClientError – 由于查询参数为空或格式无效，因此服务器拒绝了该请求，并向查看器返回了 HTTP 400 状态代码。
- RejectedByContentType – 由于在字段级加密的配置中没有指定内容类型，因此服务器拒绝了该请求，并向查看器返回了 HTTP 400 状态代码。
- RejectedByQueryArgs – 由于在字段级加密的配置中没有指定查询参数，因此服务器拒绝了该请求，并向查看器返回了 HTTP 400 状态代码。
- ServerError – 源服务器返回了错误。

如果请求超出字段级加密配额（以前称作限制），则此字段包含下列错误代码之一，并且服务器向查看器返回 HTTP 状态代码 400。有关字段级加密的当前配额的列表，请参对字段级加密的配额。

- FieldLengthLimitClientError – 配置为加密的字段超出允许的最大长度。
- FieldNumberLimitClientError – 将分配配置为加密的请求包含的字段数超过允许值。
- RequestLengthLimitClientError – 在配置了字段级加密时，请求正文的长度超出允许的最大长度。

如果没有为分配配置字段级加密，则此字段的值为连字符 (-)。

26. **fle-encrypted-fields**

服务器加密并转发到源的字段级加密字段的数量。CloudFront 服务器在加密数据时会将处理的请求传输到源，这样一来，即使 fle-status 的值为错误，此字段也会具有值。

如果没有为分配配置字段级加密，则此字段的值为连字符 (-)。

27. **c-port**

查看器发出的请求的端口号。

28.**time-to-first-byte**

从收到请求到写入响应的第一个字节之间的秒数（在服务器上测量）。

29.**x-edge-detailed-result-type**

此字段包含与 x-edge-result-type 字段相同的值，但以下情况除外：

- 当对象从源护盾层提供给查看器时，该字段包含 OriginShieldHit。
- 当对象不在 CloudFront 缓存中并且响应是由源请求 Lambda@Edge 函数生成时，此字段包含 MissGeneratedResponse。
- 当 x-edge-result-type 字段的值为 Error 时，此字段包含以下值之一，其中包含有关错误的更多信息：
  - AbortedOrigin – 服务器遇到了源方面的问题。
  - ClientCommError – 由于服务器与查看器之间的通信问题，对查看器的响应已中断。
  - ClientGeoBlocked - 将分配配置为拒绝来自查看器地理位置的请求。
  - ClientHungUpRequest - 查看器在发送请求时提前停止。
  - Error – 出现错误，其错误类型不适合任何其他类别。当服务器从缓存提供错误响应时，可能会发生此类型的错误。
  - InvalidRequest – 服务器收到了来自查看器的无效请求。
  - InvalidRequestBlocked - 阻止对所请求资源的访问。
  - InvalidRequestCertificate - 分配与建立 HTTPS 连接的 SSL/TLS 证书不匹配。
  - InvalidRequestHeader - 请求包含无效的标头。
  - InvalidRequestMethod - 未将分配配置为处理所使用的 HTTP 请求方法。当分配仅支持可缓存请求时，可能会发生这种情况。
  - OriginCommError – 连接到源或从源读取数据时，请求超时。
  - OriginConnectError – 服务器无法连接到源。
  - OriginContentRangeLengthError - 源响应中的 Content-Length 标头与 Content-Range 标头中的长度不匹配。
  - OriginDnsError – 服务器无法解析源的域名。
  - OriginError - 源返回不正确的响应。
  - OriginHeaderTooBigError - 源返回的标头太大，边缘服务器无法处理。
  - OriginInvalidResponseError - 源返回无效响应。
  - OriginReadError – 服务器无法从源读取。

- OriginWriteError – 服务器无法写入到源。
- OriginZeroSizeObjectError - 从源发送的零大小对象会导致错误。
- SlowReaderOriginError - 查看器读取导致源错误的消息时速度较慢。

30 **sc-content-type**

响应的 HTTP Content-Type 标头的值。

31 **sc-content-len**

响应的 HTTP Content-Length 标头的值。

32 **sc-range-start**

当响应包含 HTTP Content-Range 标头时，此字段包含范围起始值。

33 **sc-range-end**

当响应包含 HTTP Content-Range 标头时，此字段包含范围结束值。

以下是分配的一个日志文件示例：

```
#Version: 1.0
#Fields: date time x-edge-location sc-bytes c-ip cs-method cs(Host) cs-uri-stem sc-
status cs(Referer) cs(User-Agent) cs-uri-query cs(Cookie) x-edge-result-type x-edge-
request-id x-host-header cs-protocol cs-bytes time-taken x-forwarded-for ssl-protocol
 ssl-cipher x-edge-response-result-type cs-protocol-version fle-status fle-encrypted-
fields c-port time-to-first-byte x-edge-detailed-result-type sc-content-type sc-
content-len sc-range-start sc-range-end
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
 SOX4xwn4XV6Q4rgb7XiVGOHms_BGlTAC4KyHmureZmBNrjGdRLiNIQ== d111111abcdef8.cloudfront.net
 https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit
 text/html 78 - -
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
 k6WGMNkEzR5BEM_SaF47gjtX9zBDO2m349OY2an0QPEaUum1ZOLrow== d111111abcdef8.cloudfront.net
 https 23 0.000 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.000 Hit
 text/html 78 - -
```

```
2019-12-04 21:02:31 LAX1 392 192.0.2.100 GET d111111abcdef8.cloudfront.net /
index.html 200 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;
%20x64)%20AppleWebKit/537.36%20(KHTML,%20like
%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Hit
 f37nTMVvnKvV2ZSvEsivup_c2kZ7VXzYdjC-GUQZ5qNs-89BlWazbw== d111111abcdef8.cloudfront.net
 https 23 0.001 - TLSv1.2 ECDHE-RSA-AES128-GCM-SHA256 Hit HTTP/2.0 - - 11040 0.001 Hit
 text/html 78 - -
2019-12-13 22:36:27 SEA19-C1 900 192.0.2.200 GET d111111abcdef8.cloudfront.net /
favicon.ico 502 http://www.example.com/ Mozilla/5.0%20(Windows
%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Error
 1pkpNfBQ39sYMnjjUQjmH2w1wdJnbHYTbag21o_3OfcQgPzdL2RSSQ== www.example.com http 675
 0.102 - - - Error HTTP/1.1 - - 25260 0.102 OriginDnsError text/html 507 - -
2019-12-13 22:36:26 SEA19-C1 900 192.0.2.200 GET d111111abcdef8.cloudfront.net / 502
 - Mozilla/5.0%20(Windows%20NT%2010.0;%20Win64;%20x64)%20AppleWebKit/537.36%20(KHTML,
%20like%20Gecko)%20Chrome/78.0.3904.108%20Safari/537.36 - - Error
 3AqrZGCnF_g0-5KOvfA7c9XLcf4YGvMFSeFdIetR1N_2y8jSis8Zxg== www.example.com http 735
 0.107 - - - Error HTTP/1.1 - - 3802 0.107 OriginDnsError text/html 507 - -
2019-12-13 22:37:02 SEA19-C2 900 192.0.2.200 GET d111111abcdef8.cloudfront.net / 502
 - curl/7.55.1 - - Error kBkDzGnceVtWHqSCqBUqtA_cEs2T3tFUBbnBNkB9El_uVRhHgcZfcw==
 www.example.com http 387 0.103 - - - Error HTTP/1.1 - - 12644 0.103 OriginDnsError
 text/html 507 - -
```

标准日志的费用

标准日志记录是 CloudFront 的一个可选功能。启用标准日志记录无额外费用。但是，在 Amazon S3 上存储和访问文件（您可随时删除他们）会产生常规的 Amazon S3 费用。

有关 Amazon S3 定价的更多信息，请参阅 Amazon S3 定价。

有关 CloudFront 定价的更多信息，请参阅 CloudFront 定价。

实时日志

利用 CloudFront 实时日志，您可以实时获取有关向分配发出的请求的信息（日志在收到请求后的几秒钟内传输）。您可以使用实时日志来进行监控和分析，并根据内容交付性能采取相应措施。

CloudFront 实时日志是可配置的。您可以选择：

- 实时日志的采样率 – 即希望接收实时日志记录的请求的百分比。
- 希望在日志记录中接收的特定字段。
- 要接收实时日志的特定缓存行为（路径模式）。

CloudFront 实时日志将传送到 Amazon Kinesis Data Streams 中您选择的数据流。您可以构建自己的 Kinesis 数据流消费程序，或使用 Amazon Data Firehose 将日志数据发送到 Amazon Simple Storage Service（Amazon S3）、Amazon Redshift、Amazon OpenSearch Service（OpenSearch Service）或第三方日志处理服务。

CloudFront 除了收取因使用 Kinesis Data Streams 产生的费用外，还针对实时日志进行收费。有关定价的更多信息，请参阅 Amazon CloudFront 定价和 Amazon Kinesis Data Streams 定价。

> **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。CloudFront 将尽力提供实时日志。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。当实时日志中省略了日志条目时，实时日志中的条目数将与 AWS 账单和使用率报告中出现的使用率不匹配。

了解实时日志配置

要使用 CloudFront 实时日志，请先创建实时日志配置。实时日志配置包含有关要接收的日志字段、日志记录的采样率 以及要在其中传输日志的 Kinesis 数据流的信息。

具体而言，实时日志配置包含以下设置：

- 名称
- 采样率
- 字段
- 端点（Kinesis 数据流）
- IAM 角色

名称

用于标识实时日志配置的名称。

采样率

采样率是一个介于 1 和 100 之间的整数（含 1 和 100），用于确定作为实时日志记录发送到 Kinesis Data Streams 的查看器请求的百分比。要在实时日志中包含每个查看器请求，请指定 100 作为采样率。可以选择较低的采样率来降低成本，同时仍在实时日志中接收具有代表性的请求数据示例。

字段

每个实时日志记录中包含的字段列表。每个日志记录最多可包含 40 个字段，您可以选择接收所有可用字段，也可以选择仅接收监控和分析性能所需的字段。

以下列表包含每个字段名称和该字段中信息的说明。字段将按照它们在传输到 Kinesis Data Streams 的日志记录中的显示顺序列出。

字段 46-63 是常见的媒体客户端数据（CMCD），媒体播放器客户端可以在每次请求时将其发送到 CDN。您可以使用这些数据来了解每个请求，例如媒体类型（音频、视频）、播放速率和直播时长。只有在将这些字段发送到 CloudFront 后，它们才会显示在您的实时日志中。

1. **timestamp**

   边缘服务器完成对请求的响应的日期和时间。

2. **c-ip**

   已发出请求的查看器的 IP 地址，例如 `192.0.2.183` 或 `2001:0db8:85a3::8a2e:0370:7334`。如果查看器已使用 HTTP 代理或负载均衡器发送请求，则此字段的值为该代理或负载均衡器的 IP 地址。另请参阅 `x-forwarded-for` 字段。

3. **time-to-first-byte**

   从收到请求到写入响应的第一个字节之间的秒数（在服务器上测量）。

4. **sc-status**

   服务器响应的 HTTP 状态代码（例如 `200`）。

5. **sc-bytes**

   服务器在响应请求时向查看器提供的字节的总数，包括标头。对于 WebSocket 连接，这是通过连接从服务器发送到客户端的字节的总数。

6. **cs-method**

   从查看器接收的 HTTP 请求方法。

7. **cs-protocol**

   查看器请求的协议（`http`、`https`、`ws` 或 `wss`）。

8. **cs-host**

查看器在该请求的 Host 标头中包含的值。如果您在对象 URL 中使用 CloudFront 域名（如 d111111abcdef8.cloudfront.net），则此字段将包含该域名。如果在您的对象 URL（例如 www.example.com）中使用的是备用域名（CNAME），则此字段将包含备用域名。

9. **cs-uri-stem**

   整个请求 URL，包括查询字符串（如果有），但不包括域名。例如，/images/cat.jpg?mobile=true。

   > **Note**
   >
   > 在标准日志中，该 cs-uri-stem 值不包括查询字符串。

10. **cs-bytes**

    查看器包含在请求中的数据字节总数，包括标头。对于 WebSocket 连接，这是通过连接从客户端发送到服务器的字节的总数。

11. **x-edge-location**

    服务请求的边缘站点。每个边缘站点由三个字母的代码和任意分配的数字来确定（例如，DFW3）。三个字母代码通常对应邻近边缘站点的地理位置的机场的国际航空协会（IATA）机场代码。（这些缩写将来可能会更改。）

12. **x-edge-request-id**

    唯一地标识请求的不透明字符串。CloudFront 还会在 x-amz-cf-id 响应标头中发送此字符串。

13. **x-host-header**

    CloudFront 分配的域名（例如，d111111abcdef8.cloudfront.net）。

14. **time-taken**

    服务器收到查看器的请求的时间与服务器将响应的最后一个字节写入输出队列的时间之间相隔的秒数（精确至千分之一秒，例如 0.082），以服务器上测量的时间为准。从查看器的角度看，由于网络延迟和 TCP 缓冲的原因，获得完整响应的总时间将会超过该值。

15. **cs-protocol-version**

    查看器在请求中指定的 HTTP 版本。可能的值包括 HTTP/0.9、HTTP/1.0、HTTP/1.1、HTTP/2.0 和 HTTP/3.0。

16. **c-ip-version**

请求的 IP 版本（IPv4 或 IPv6）。

17. **cs-user-agent**

    请求中的 User-Agent 标头的值。User-Agent 标头标识请求的来源，例如提交请求的设备和浏览器的类型，如果请求来自搜索引擎，则标识具体的搜索引擎。

18. **cs-referer**

    请求中的 Referer 标头的值。这是发出请求的域的名称。常见引用站点包括搜索引擎、直接链接到您的对象的其他网站及您自己的网站。

19. **cs-cookie**

    请求中的 Cookie 标头，包括名称/值对和关联的属性。

    > **Note**
    >
    > 此字段被截断为 800 字节。

20. **cs-uri-query**

    请求 URL 的查询字符串部分（如果有）。

21. **x-edge-response-result-type**

    服务器在将响应返回到查看器之前如何对响应进行分类。另请参阅 x-edge-result-type 字段。可能的值包括：

    - Hit – 服务器从缓存中将对象提供给查看器。
    - RefreshHit – 服务器在缓存中找到了对象，但该对象已过期，因此，服务器联系了源，以验证缓存是否具有该对象的最新版本。
    - Miss – 缓存中的对象无法满足请求，因此，服务器将请求转发到源服务器并将结果返回到查看器。
    - LimitExceeded – 请求被拒绝，因为超出了 CloudFront 配额（以前称为限制）。
    - CapacityExceeded – 服务器返回了 503 错误，因为它在请求时没有足够的容量来服务对象。
    - Error – 通常，这意味着请求会导致客户端错误（sc-status 字段的值在 4xx 范围内）或服务器错误（sc-status 字段的值在 5xx 范围内）。

      如果 x-edge-result-type 字段的值为 Error，而此字段的值不是 Error，则客户端在下载完成前已断开连接。

- `Redirect` – 服务器已根据分发设置将查看器从 HTTP 重定向到 HTTPS。

22 **`x-forwarded-for`**

如果查看器已使用 HTTP 代理或负载均衡器发送请求，则 `c-ip` 字段的值为该代理或负载均衡器的 IP 地址。在本例中，此字段为发出请求的查看器的 IP 地址。此字段可以包含多个逗号分隔的 IP 地址。每个 IP 地址可以是 IPv4 地址（如 `192.0.2.183`）或 IPv6 地址（如 `2001:0db8:85a3::8a2e:0370:7334`）。

23 **`ssl-protocol`**

如果请求使用了 HTTPS，则此字段包含查看器和服务器协商用于传输请求和响应的 SSL/TLS 协议。有关可能的值的列表，请参阅查看器和 CloudFront 之间支持的协议和密码中支持的 SSL/TLS 协议。

24 **`ssl-cipher`**

如果请求使用了 HTTPS，则此字段包含查看器和服务器协商用于加密请求和响应的 SSL/TLS 密码。有关可能的值的列表，请参阅查看器和 CloudFront 之间支持的协议和密码中支持的 SSL/TLS 密码。

25 **`x-edge-result-type`**

在最后一个字节离开服务器后服务器如何对响应进行分类。在某些情况下，结果类型可能会在服务器准备发送响应的时间与完成发送响应的时间之间发生变化。另请参阅 `x-edge-response-result-type` 字段。

例如，在 HTTP 流中，假设服务器在缓存中发现流的一个区段。在这种情况下，此字段的值通常为 `Hit`。但是，如果查看器在服务器传送整个区段之前关闭连接，则最终结果类型（以及此字段的值）为 `Error`。

WebSocket 连接将具有此字段的值 `Miss`，因为内容不可缓存，并直接通过代理回到源。

可能的值包括：

- `Hit` – 服务器从缓存中将对象提供给查看器。
- `RefreshHit` – 服务器在缓存中找到了对象，但该对象已过期，因此，服务器联系了源，以验证缓存是否具有该对象的最新版本。
- `Miss` – 缓存中的对象无法满足请求，因此，服务器将请求转发到源并将结果返回到查看器。
- `LimitExceeded` – 请求被拒绝，因为超出了 CloudFront 配额（以前称为限制）。
- `CapacityExceeded` – 服务器返回了 HTTP 503 状态代码，因为它在请求时没有足够的容量来服务对象。

- Error – 通常，这意味着请求会导致客户端错误（sc-status 字段的值在 4xx 范围内）或服务器错误（sc-status 字段的值在 5xx 范围内）。如果 sc-status 字段的值为 200，或者如果此字段的值为 Error 且 x-edge-response-result-type 字段的值不是 Error，则表示 HTTP 请求已成功，但客户端在接收所有字节之前断开连接。
- Redirect – 服务器已根据分发设置将查看器从 HTTP 重定向到 HTTPS。

26. **fle-encrypted-fields**

服务器加密并转发到源的字段级加密字段的数量。CloudFront 服务器在加密数据时会将处理的请求传输到源，这样一来，即使 fle-status 的值为错误，此字段也会具有值。

27. **fle-status**

在为分配配置字段级加密时，此字段包含一个指示是否已成功处理请求正文的代码。如果服务器成功处理了请求正文，加密了指定字段中的值并将请求转发到源，则此字段的值为 Processed。在这种情况下，x-edge-result-type 的值仍可以指示客户端或服务器端错误。

此字段的可能值包括：

- ForwardedByContentType – 由于没有配置内容类型，因此服务器将请求转发到了源，而不进行解析或加密。
- ForwardedByQueryArgs – 由于请求包含的查询参数不在字段级加密的配置中，因此服务器将请求转发到了源，而不进行解析或加密。
- ForwardedDueToNoProfile – 由于在字段级加密的配置中没有指定配置文件，因此服务器将请求转发到了源，而不进行解析或加密。
- MalformedContentTypeClientError – 由于 Content-Type 标头值的格式无效，因此，服务器拒绝了该请求并向查看器返回了 HTTP 400 状态代码。
- MalformedInputClientError – 由于请求正文的格式无效，因此服务器拒绝了该请求，并向查看器返回了 HTTP 400 状态代码。
- MalformedQueryArgsClientError – 由于查询参数为空或格式无效，因此服务器拒绝了该请求，并向查看器返回了 HTTP 400 状态代码。
- RejectedByContentType – 由于在字段级加密的配置中没有指定内容类型，因此服务器拒绝了该请求，并向查看器返回了 HTTP 400 状态代码。
- RejectedByQueryArgs – 由于在字段级加密的配置中没有指定查询参数，因此服务器拒绝了该请求，并向查看器返回了 HTTP 400 状态代码。
- ServerError – 源服务器返回了错误。

如果请求超出字段级加密配额（以前称作限制），则此字段包含下列错误代码之一，并且服务器向查看器返回 HTTP 状态代码 400。有关字段级加密的当前配额的列表，请参对字段级加密的配额。

- FieldLengthLimitClientError – 配置为加密的字段超出允许的最大长度。
- FieldNumberLimitClientError – 将分配配置为加密的请求包含的字段数超过允许值。
- RequestLengthLimitClientError – 在配置了字段级加密时，请求正文的长度超出允许的最大长度。

28 **sc-content-type**

响应的 HTTP Content-Type 标头的值。

29 **sc-content-len**

响应的 HTTP Content-Length 标头的值。

30 **sc-range-start**

当响应包含 HTTP Content-Range 标头时，此字段包含范围起始值。

31 **sc-range-end**

当响应包含 HTTP Content-Range 标头时，此字段包含范围结束值。

32 **c-port**

查看器发出的请求的端口号。

33 **x-edge-detailed-result-type**

此字段包含与 x-edge-result-type 字段相同的值，但以下情况除外：

- 当对象从源护盾层提供给查看器时，该字段包含 OriginShieldHit。
- 当对象不在 CloudFront 缓存中并且响应是由源请求 Lambda@Edge 函数生成时，此字段包含 MissGeneratedResponse。
- 当 x-edge-result-type 字段的值为 Error 时，此字段包含以下值之一，其中包含有关错误的更多信息：
  - AbortedOrigin – 服务器遇到了源方面的问题。
  - ClientCommError – 由于服务器与查看器之间的通信问题，对查看器的响应已中断。
  - ClientGeoBlocked - 将分配配置为拒绝来自查看器地理位置的请求。
  - ClientHungUpRequest - 查看器在发送请求时提前停止。

- Error – 出现错误，其错误类型不适合任何其他类别。当服务器从缓存提供错误响应时，可能会发生此类型的错误。
- InvalidRequest – 服务器收到了来自查看器的无效请求。
- InvalidRequestBlocked - 阻止对所请求资源的访问。
- InvalidRequestCertificate - 分配与建立 HTTPS 连接的 SSL/TLS 证书不匹配。
- InvalidRequestHeader - 请求包含无效的标头。
- InvalidRequestMethod - 未将分配配置为处理所使用的 HTTP 请求方法。当分配仅支持可缓存请求时，可能会发生这种情况。
- OriginCommError – 连接到源或从源读取数据时，请求超时。
- OriginConnectError – 服务器无法连接到源。
- OriginContentRangeLengthError - 源响应中的 Content-Length 标头与 Content-Range 标头中的长度不匹配。
- OriginDnsError – 服务器无法解析源的域名。
- OriginError - 源返回不正确的响应。
- OriginHeaderTooBigError - 源返回的标头太大，边缘服务器无法处理。
- OriginInvalidResponseError - 源返回无效响应。
- OriginReadError – 服务器无法从源读取。
- OriginWriteError – 服务器无法写入到源。
- OriginZeroSizeObjectError - 从源发送的零大小对象会导致错误。
- SlowReaderOriginError - 查看器读取导致源错误的消息时速度较慢。

34.**c-country**

表示查看器的地理位置的国家/地区代码，由查看器的 IP 地址决定。有关国家/地区代码的列表，请参阅 ISO 3166-1 alpha-2。

35.**cs-accept-encoding**

查看器请求中的 Accept-Encoding 标头的值。

36.**cs-accept**

查看器请求中的 Accept 标头的值。

37.**cache-behavior-path-pattern**

标识与查看器请求匹配的缓存行为的路径模式。

38. **cs-headers**

查看器请求中的 HTTP 标头（名称和值）。

> **Note**
>
> 此字段被截断为 800 字节。

39. **cs-header-names**

查看器请求中的 HTTP 标头的名称（而不是值）。

> **Note**
>
> 此字段被截断为 800 字节。

40. **cs-headers-count**

查看器请求中的 HTTP 标头的数量。

41. **origin-fbl**

CloudFront 与您的源之间第一字节延迟的秒数。

42. **origin-lbl**

CloudFront 与您的源之间最后一个字节延迟的秒数。

43. **asn**

包含查看器的自治系统号 (ASN)。

44. **primary-distribution-id**

启用持续部署后，此 ID 将标识当前分配中的哪个分配是主分配。

45. **primary-distribution-dns-name**

启用持续部署后，此值显示与当前 CloudFront 分配相关的主域名（例如 d111111abcdef8.cloudfront.net）。

> **实时日志中的 CMCD 字段**
>
> 有关这些字段的更多信息，请参阅 CTA 规范 Web 应用程序视频生态系统 - 通用媒体客户端数据 CTA-5004 文档。

46. **cmcd-encoded-bitrate**

    请求的音频或视频对象的编码比特率。

47. **cmcd-buffer-length**

    所请求媒体对象的缓冲区长度。

48. **cmcd-buffer-starvation**

    缓冲区是否在先前的请求和对象请求之间的某个时刻被耗尽。这可能会导致播放器处于重新缓冲状态，从而导致视频或音频播放停滞。

49. **cmcd-content-id**

    标识当前内容的唯一字符串。

50. **cmcd-object-duration**

    请求对象的播放时长（以毫秒为单位）。

51. **cmcd-deadline**

    从请求时间算起的截止日期，在这一时间，该对象的第一个样本必须可用，这样可以避免缓冲区处于欠载状态或出现其他播放问题。

52. **cmcd-measured-throughput**

    客户端和服务器之间的吞吐量，由客户端进行测量。

53. **cmcd-next-object-request**

    所请求下一个对象的相对路径。

54. **cmcd-next-range-request**

    如果下一个请求是部分对象请求，则此字符串表示要请求的字节范围。

55. **cmcd-object-type**

    所请求的当前对象的媒体类型。

56. **cmcd-playback-rate**

    如果是实时播放，则为 1；如果是倍速播放，则为 2；如果不播放，则为 0。

57. **cmcd-requested-maximum-throughput**

    请求的最大吞吐量，客户端认为足以交付资源。

58. **cmcd-streaming-format**

    定义当前请求的流媒体格式。

59. **cmcd-session-id**

    标识当前播放会话的 GUID。

60. **cmcd-stream-type**

    令牌识别区段的可用性。 v = 所有区段都可用。l = 区段会随着时间的推移而变得可用。

61. **cmcd-startup**

    如果在启动、搜索或缓冲区空事件后的恢复期间迫切需要该对象，则可包含不带值的键。

62. **cmcd-top-bitrate**

    客户端可以播放的最高比特率副本。

63. **cmcd-version**

    本规范的版本，用于解释已定义的键名和键值。如果省略此键，则客户端和服务器必须 将这些值解释为由版本 1 定义。

端点（Kinesis 数据流）

端点包含有关您在其中发送实时日志的 Kinesis 数据流的信息。您需要提供数据流的 Amazon 资源名称（ARN）。

有关创建 Kinesis 数据流的更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的以下主题。

- 使用控制台管理流
- 使用 AWS CLI 执行基本 Kinesis 数据流操作
- 创建流（使用AWS SDK for Java）

创建数据流时，您需要指定分片的数量。使用以下信息可帮助您估计所需的分片数量。

估计 Kinesis 数据流的分片数

1. 计算（或估算）您的 CloudFront 分配每秒钟接收的请求数。

   您可以使用 CloudFront 使用情况报告（在 CloudFront 控制台中）和 CloudFront 指标（在 CloudFront 和 Amazon CloudWatch 控制台中）来帮助您计算每秒的请求数。

2. 确定单个实时日志记录的典型大小。

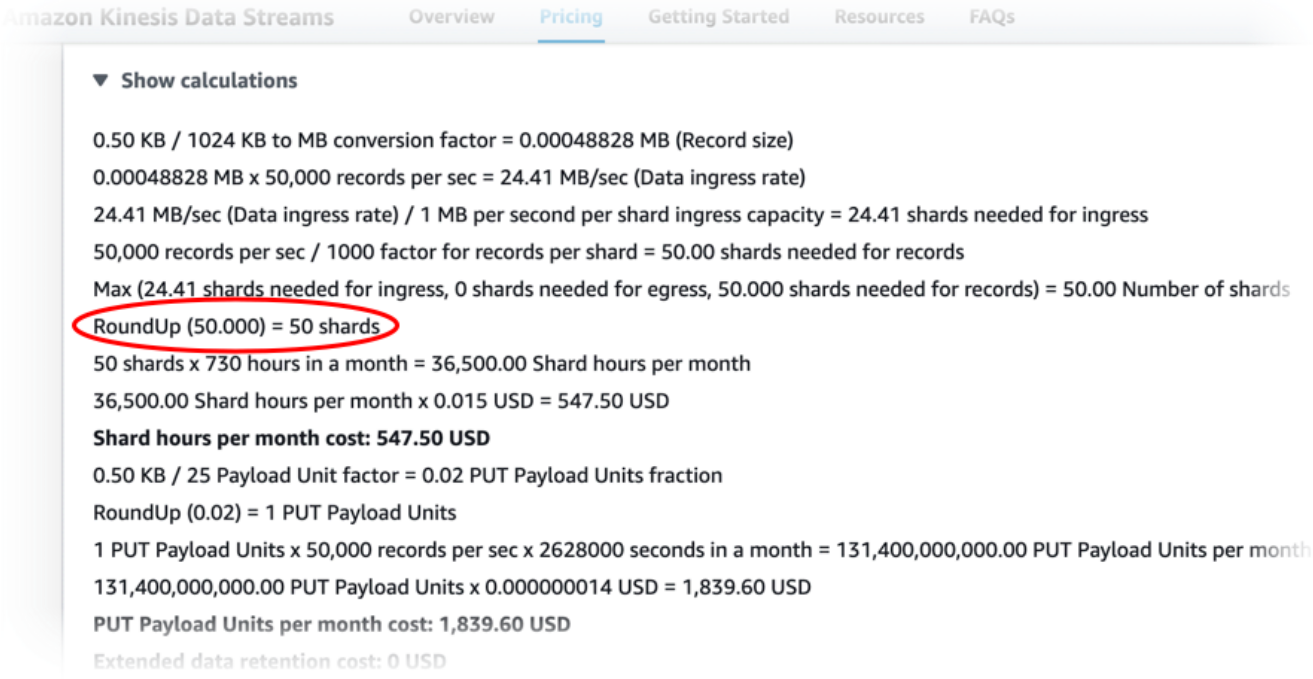
   通常，单个日志记录约为 500 字节。一个包含所有可用字段的大型记录一般约为 1KB。

   如果您不确定您的日志记录的大小，可以启用低采样率（例如 1%）的实时日志，然后使用 Kinesis Data Streams 中的监测数据计算记录的平均大小（记录总数除以总的传入字节数）。

3. 在 Amazon Kinesis Data Streams 定价页面上的定价计算器中，输入每秒的请求（记录）数，以及单个日志记录的平均记录大小。然后选择显示计算。

   定价计算器会显示您需要的分区数量。（它还向您显示估算费用。）

   以下示例显示，对于 0.5KB 的平均记录大小和每秒 50000 个请求，您需要 50 个分区。



IAM 角色

AWS Identity and Access Management (IAM) 角色，授予 CloudFront 向您的 Kinesis 数据流传送实时日志的权限。

使用 CloudFront 控制台创建实时日志配置时，可以选择创建新服务角色以让控制台为您创建 IAM 角色。

当您使用 AWS CloudFormation 或 CloudFront API（AWS CLI 或软件开发工具包）创建实时日志配置时，必须自行创建 IAM 角色并提供角色 ARN。要自行创建 IAM 角色，请使用以下策略。

IAM 角色信任策略

要使用以下 IAM 角色信任策略，请将 *111122223333* 替换为您的 AWS 账户号。此策略中的 `Condition` 条件元素有助于防止出现混淆代理人问题，因为 CloudFront 只能代表 AWS 账户中的分配承担此角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cloudfront.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

未加密数据流的 IAM 角色权限策略

要使用以下策略，请将 *arn:aws:kinesis:us-east-2:123456789012:stream/StreamName* 替换为 Kinesis 数据流的 ARN。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream",
```

```
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-2:123456789012:stream/StreamName"
            ]
        }
    ]
}
```

已加密数据流的 IAM 角色权限策略

要使用以下策略，请用 Kinesis 数据流的 ARN 替换 *arn:aws:kinesis:us-east-2:123456789012:stream/StreamName*，以及用 AWS KMS key 的 ARN 替换 *arn:aws:kms:us-east-2:123456789012:key/e58a3d0b-fe4f-4047-a495-ae03cc73d486*。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream",
                "kinesis:PutRecord",
                "kinesis:PutRecords"
            ],
            "Resource": [
                "arn:aws:kinesis:us-east-2:123456789012:stream/StreamName"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-2:123456789012:key/e58a3d0b-fe4f-4047-a495-
ae03cc73d486"
            ]
        }
    ]
```

```
}
```

创建和使用实时日志配置

可以使用实时日志配置实时获取有关向分配发出的请求的信息（日志在收到请求后的几秒钟内传输）。您可以使用 AWS Command Line Interface (AWS CLI) 或 CloudFront API 在 CloudFront 控制台中创建实时日志配置。

要使用实时日志配置，请将它附加到 CloudFront 分配中的一个或多个缓存行为。

创建实时日志配置（控制台）

创建实时日志配置

1. 登录 AWS Management Console 并通过以下网址在 CloudFront 控制台中打开日志页面：https://console.aws.amazon.com/cloudfront/v4/home?#/logs。
2. 选择实时配置选项卡。
3. 选择创建配置。
4. 对于名称，请输入配置名称。
5. 对于采样率，请输入您希望接收其日志记录的请求的百分比。
6. 对于字段，请选择要在实时日志中接收的字段。
   - 要在日志中包含所有 CMCD 字段，请选择 CMCD 所有键。
7. 对于端点，请选择一个或多个 Kinesis 数据流来接收实时日志。

   > **Note**
   >
   > CloudFront 实时日志将传送到您在 Kinesis Data Streams 中指定的数据流。要读取和分析实时日志，您可以构建自己的 Kinesis 数据流使用者。您也可以使用 Firehose 将日志数据发送到 Amazon S3、Amazon Redshift、Amazon OpenSearch Service 或第三方日志处理服务。

8. 对于 IAM 角色，请选择创建新服务角色或选择现有角色。您必须具有创建 IAM 角色的权限。
9. （可选）对于分配，请选择要附加到实时日志配置的 CloudFront 分配和缓存行为。
10. 选择创建配置。

如果成功，控制台将显示您刚创建的实时日志配置的详细信息。

有关更多信息，请参阅 了解实时日志配置。

创建实时日志配置 (AWS CLI)

要使用 AWS Command Line Interface (AWS CLI) 创建实时日志配置，请使用 aws cloudfront create-realtime-log-config 命令。您可以使用输入文件来提供命令的输入参数，而不是将每个单独的参数指定为命令行输入。

创建实时日志配置（带输入文件的 CLI）

1. 使用以下命令创建名为 `rtl-config.yaml` 的文件，其中包含 create-realtime-log-config 命令的所有输入参数。

```
aws cloudfront create-realtime-log-config --generate-cli-skeleton yaml-input > rtl-config.yaml
```

2. 打开刚创建的名为 `rtl-config.yaml` 的文件。编辑该文件以指定所需的实时日志配置设置，然后保存该文件。请注意以下几点：

   - 对于 `StreamType`，唯一的有效值为 `Kinesis`。

   有关实时长配置设置的更多信息，请参阅了解实时日志配置。

3. 使用以下命令通过 `rtl-config.yaml` 文件中的输入参数创建实时日志配置。

```
aws cloudfront create-realtime-log-config --cli-input-yaml file://rtl-config.yaml
```

如果成功，命令的输出将显示您刚创建的实时日志配置的详细信息。

将实时日志配置附加到现有分配（带输入文件的 CLI）

1. 使用以下命令保存要更新的 CloudFront 分配的分配配置。将 *distribution_ID* 替换为分配的 ID。

```
aws cloudfront get-distribution-config --id distribution_ID --output yaml > dist-config.yaml
```

2. 打开刚创建的名为 `dist-config.yaml` 的文件。编辑该文件，对要更新的每个缓存行为进行以下更改以使用实时日志配置。

   - 在缓存行为中，添加名为 `RealtimeLogConfigArn` 的字段。对于字段的值，使用要附加到此缓存行为的实时日志配置的 ARN。
   - 将 `ETag` 字段重命名为 `IfMatch`，但不更改字段的值。

   完成后保存该文件。

3. 使用以下命令更新分配以使用实时日志配置。将 *`distribution_ID`* 替换为分配的 ID。

```
aws cloudfront update-distribution --id distribution_ID --cli-input-yaml file://
dist-config.yaml
```

如果成功，命令的输出将显示您刚刚更新的分配的详细信息。

创建实时日志配置 (API)

要使用 CloudFront API 创建实时日志配置，请使用 CreateRealtimeLogConfig。有关您在此 API 调用中指定的参数的更多信息，请参阅 了解实时日志配置 以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

创建实时日志配置后，可以使用下列 API 调用之一将该配置附加到缓存行为：

- 要将该配置附加到现有分配中的缓存行为，请使用 UpdateDistribution。
- 要将该配置附加到新分配中的缓存行为，请使用 CreateDistribution。

对于这两个 API 调用，请在缓存行为内的 `RealtimeLogConfigArn` 字段中提供实时日志配置的 ARN。有关您在这些 API 调用中指定的其他字段的更多信息，请参阅 分配设置参考 以及有关 AWS SDK 或其他 API 客户端的 API 参考文档。

创建 Kinesis Data Streams 使用者

要读取和分析实时日志，您可以构建或使用 Kinesis Data Streams 使用者。为 CloudFront 实时日志构建使用器时，务必了解每个实时日志记录中的字段始终以相同的顺序传递（如字段部分中所列）。确保构建您的使用器以适应此固定顺序。

例如，假设实时日志配置只包含以下三个字段：time-to-first-byte、sc-status 和 c-country。在此情况下，最后一个字段 c-country 始终是每个日志记录中的字段编号 3。但是，如果您稍后将字段添加到实时日志配置中，则记录中每个字段的位置可能会发生更改。

例如，如果您将 sc-bytes 和 time-taken 字段添加到实时日志配置，则根据字段部分中显示的顺序将这些字段插入到每个日志记录中。所有五个字段的最终顺序为 time-to-first-byte、sc-status、sc-bytes、time-taken 和 c-country。c-country 字段最初是字段编号 3，但现在为字段编号 5。如果您将字段添加到实时日志配置中，请确保您的使用器应用程序可以处理更改日志记录中位置的字段。

问题排查实时日志

创建实时日志配置后，您可能会发现未将任何记录或所有记录传输到 Kinesis Data Streams。在此情况下，您应先验证您的 CloudFront 分配是否正在接收查看器请求。如果是这样，则可以检查以下设置以继续进行问题排查。

IAM 角色权限

为了将实时日志记录传输到 Kinesis 数据流，CloudFront 将使用实时日志配置中的 IAM 角色。请确保角色信任策略和角色权限策略与 IAM 角色 中显示的策略匹配。

Kinesis Data Streams 限制

如果 CloudFront 将实时日志记录写入您的 Kinesis 数据流的速度超过流的处理速度，Kinesis Data Streams 可能会限制来自 CloudFront 的请求。在此情况下，您可以增加 Kinesis 数据流中的分片数量。每个分片最多可以支持每秒写入 1000 条记录，最大数据写入数为每秒 1 MB。

边缘函数日志

可以使用 Amazon CloudWatch Logs 获取边缘函数（包括 Lambda@Edge 和 CloudFront Functions）的日志。使用 CloudWatch 控制台或 CloudWatch Logs API 访问日志。

> **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。CloudFront 将尽力提供边缘函数日志。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。当边缘函数日志中省略了日志条目时，边缘函数日志中的条目数将与 AWS 账单和使用率报告中出现的使用率不匹配。

Lambda@Edge 日志

Lambda@Edge 会自动将函数日志发送到 CloudWatch Logs，在函数运行的 AWS 区域中创建日志流。日志组名称的格式为 `/aws/lambda/us-east-1.`*`function-name`*，其中 *`function-name`* 是您在创建函数时为其指定的名称，`us-east-1` 是在其中创建函数的 AWS 区域的区域代码。日志组名称始终包含 `us-east-1`，即使对于运行函数的其他区域的日志组也是如此。

> **Note**
>
> Lambda@Edge 会基于请求量和日志大小来限制日志。

您必须检查正确的 AWS 区域中的 CloudWatch 日志文件，才能看到 Lambda@Edge 函数日志文件。要查看 Lambda@Edge 函数运行的区域，请在 CloudFront 控制台上查看此函数的指标图表。指标针对各个 AWS 区域显示。在同一页上，您可以选择一个区域，然后查看该区域的日志文件以调查问题。

要了解有关如何将 CloudWatch Logs 与 Lambda@Edge 函数结合使用的更多信息，请参阅以下内容：

- 有关在 CloudFront 控制台的监控部分中查看图表的更多信息，请参阅the section called “使用 Amazon CloudWatch 监控 CloudFront 指标”。
- 有关向 CloudWatch Logs 发送数据所需的权限的信息，请参阅the section called “设置 IAM 权限和角色”。
- 有关向 Lambda@Edge 函数添加日志记录的信息，请参阅《AWS Lambda 开发人员指南》 中的 Node.js 中的 AWS Lambda 函数日志记录或 Python 中的 AWS Lambda 函数日志记录。
- 有关 CloudWatch 日志配额（以前称为限制）的信息，请参阅《Amazon CloudWatch Logs 用户指南》中的 CloudWatch Logs 配额。

CloudFront Functions 日志

如果 CloudFront 函数的代码包含 `console.log()` 语句，CloudFront Functions 会将这些日志行自动发送到 CloudWatch Logs。如果没有 `console.log()` 语句，则不会发送任何内容到 CloudWatch Logs 中。

CloudFront Functions 始终在美国东部（弗吉尼亚州北部）区域 (`us-east-1`) 中创建日志流，无论哪个边缘站点运行该函数。日志组名称的格式为 `/aws/cloudfront/function/`*`FunctionName`*，其中 *`FunctionName`* 是您在创建函数时为函数指定的名称。日志流名称的格式为 `YYYY/M/D/`*`UUID`*。

下面显示了发送到 CloudWatch Logs 的示例日志消息。每行都以唯一标识 CloudFront 请求的 ID 开头。消息以包含 CloudFront 分配 ID 的 START 行开头，以 END 行结尾。START 与 END 行之间是 console.log() 语句在函数中生成的日志行。

```
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== START DistributionID:
 E3E5D42GADAXZZ
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== Example function log output
U7b4hR_RaxMADupvKAvr8_m9gsGXvioUggLV5Oyq-vmAtH8HADpjhw== END
```

> **Note**
>
> CloudFront Functions 仅为处理生产请求和响应而运行的 LIVE 阶段中的函数向 CloudWatch 发送日志。测试函数时，CloudFront 不会向 CloudWatch 发送任何日志。测试输出包含有关错误、计算利用率和函数日志（console.log() 语句）的信息，但这些信息不会发送到 CloudWatch。

CloudFront Functions 使用 AWS Identity and Access Management (IAM) 服务相关角色将日志发送到您的账户中的 CloudWatch Logs。服务相关角色是一种与 AWS 服务直接关联的 IAM 角色。服务相关角色是由服务预定义的，具有服务代表您调用其他 AWS 服务所需的所有权限。CloudFront Functions 使用名为 AWSServiceRoleForCloudFrontLogger 的服务相关角色。有关此角色的更多信息，请参阅the section called “Lambda@Edge 的服务相关角色”（Lambda@Edge 使用同一个服务相关角色）。

当函数失败并出现验证错误或执行错误时，信息将记录在 CloudFront 的标准日志和实时日志中。有关错误的信息记录在 x-edge-result-type、x-edge-response-result-type 和 x-edge-detailed-result-type 字段中。

使用 AWS CloudTrail 记录 Amazon CloudFront API 调用

CloudFront 可与 AWS CloudTrail 集成，后者是一项可提供用户、角色或 AWS 服务所执行操作记录的服务。CloudTrail 将对 CloudFront 的所有 API 调用作为事件捕获。捕获的调用包括来自 CloudFront 控制台的调用和对 CloudFront API 操作的代码调用。借助 CloudTrail 收集的信息，您可以确定向 CloudFront 发出的请求、发出请求的 IP 地址、请求的发出时间及其他详细信息。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是否代表 IAM Identity Center 用户发出。

- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务 发出。

当您创建账户并可以自动访问 CloudTrail 事件历史记录时，CloudTrail 在您的 AWS 账户 中处于活动状态。CloudTrail 事件历史记录提供对 AWS 区域 中过去 90 天的已记录管理事件的可查看、可搜索、可下载和不可变记录。有关更多信息，请参见《AWS CloudTrail 用户指南》的 使用 CloudTrail 事件历史记录。查看事件历史记录不会收取 CloudTrail 费用。

要持续记录您的 AWS 账户 过去 90 天的事件，请创建跟踪或 CloudTrail Lake 事件数据存储。

CloudTrail 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。使用 AWS Management Console 创建的所有跟踪均具有多区域属性。您可以通过使用 AWS CLI 创建单区域或多区域跟踪。建议创建多区域跟踪，因为您可记录您账户中的所有 AWS 区域 的活动。如果您创建单区域跟踪，则只能查看跟踪的 AWS 区域 中记录的事件。有关跟踪的更多信息，请参阅《AWS CloudTrail 用户指南》中的为您的 AWS 账户 创建跟踪和为组织创建跟踪。

通过创建跟踪，您可以从 CloudTrail 免费向您的 Amazon S3 存储桶传送一份正在进行的管理事件的副本，但会收取 Amazon S3 存储费用。有关 CloudTrail 定价的更多信息，请参阅 AWS CloudTrail 定价。有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

CloudTrail Lake 事件数据存储

CloudTrail Lake 允许您对事件运行基于 SQL 的查询。CloudTrail Lake 可将基于行的 JSON 格式的现有事件转换为 Apache ORC 格式。ORC 是一种针对快速检索数据进行优化的列式存储格式。事件将被聚合到事件数据存储中，它是基于您通过应用高级事件选择器选择的条件的不可变的事件集合。应用于事件数据存储的选择器用于控制哪些事件持续存在并可供您查询。有关 CloudTrail Lake 的更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 AWS CloudTrail Lake。

CloudTrail Lake 事件数据存储和查询会产生费用。创建事件数据存储时，您可以选择要用于事件数据存储的定价选项。定价选项决定了摄取和存储事件的成本，以及事件数据存储的默认和最长保留期。有关 CloudTrail 定价的更多信息，请参阅 AWS CloudTrail 定价。

> **Note**
>
> CloudFront 是一项全球性服务。CloudTrail 可记录美国东部（弗吉尼亚州北部）区域的 CloudFront 事件。有关更多信息，请参阅《AWS CloudTrail 用户指南》中的全球服务事件。

如果您通过 AWS Security Token Service 使用临时安全证书，则对区域端点（如 us-west-2）的调用将在 CloudTrail 中记录到相应的区域。
有关 CloudFront 端点的更多信息，请参阅《AWS 一般参考》中的 CloudFront 端点和配额。

CloudTrail 中的 CloudFront 数据事件

数据事件可提供对资源或在资源中所执行资源操作（例如，读取或写入 CloudFront 分配）的相关信息。这些也称为数据层面操作。数据事件通常是高容量活动。默认情况下，CloudTrail 不记录数据事件。CloudTrail 事件历史记录不记录数据事件。

记录数据事件将收取额外费用。有关 CloudTrail 定价的更多信息，请参阅 AWS CloudTrail 定价。

您可以使用 CloudTrail 控制台、AWS CLI 或 CloudTrail API 操作记录 CloudFront 资源类型的数据事件。有关如何记录数据事件的更多信息，请参阅《AWS CloudTrail 用户指南》中的使用 AWS Management Console 记录数据事件和使用 AWS Command Line Interface 记录数据事件。

下表列出了您可以为其记录数据事件的 CloudFront 资源类型。数据事件类型（控制台）列显示可从 CloudTrail 控制台上的数据事件类型列表中选择的值。resources.type 值列显示了您在使用 AWS CLI 或 CloudTrail API 配置高级事件选择器时需要指定的 `resources.type` 值。记录到 CloudTrail 的数据 API 列显示了针对该资源类型记录到 CloudTrail 的 API 调用。

数据事件类型（控制台）	resources.type 值	记录至 CloudTrail 的数据 API
CloudFront KeyValueStore	`AWS::CloudFront::KeyValueStore`	• DeleteKeys • DescribeKeyValueStore • GetKey • ListKeys • PutKeys • UpdateKeys

您可以将高级事件选择器配置为在 `eventName`、`readOnly` 和 `resources.ARN` 字段上进行筛选，从而仅记录那些对您很重要的事件。有关这些字段的更多信息，请参阅《AWS CloudTrail API 参考》中的 AdvancedFieldSelector。

CloudTrail 中的 CloudFront 管理事件

管理事件提供对您的 AWS 账户内资源所执行管理操作的相关信息。这些也称为控制层面操作。默认情况下，CloudTrail 会记录管理事件。

Amazon CloudFront 将所有 CloudFront 控制面板操作记录为管理事件。要查看 CloudFront 记录到 CloudTrail 中的 Amazon CloudFront 控制面板操作列表，请参阅 Amazon CloudFront API 参考。

CloudFront 事件示例

一个事件表示一个来自任何源的请求，包括有关所请求的 API 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，因此事件不会按任何特定顺序显示。

目录

例如：UpdateDistribution

下面的示例显示了一个 CloudTrail 事件，该事件说明了 UpdateDistribution 操作。

对于对 CloudFront API 的调用，`eventSource` 是 `cloudfront.amazonaws.com`。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:role-session-name",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/role-session-name",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
```

```
                "creationDate": "2024-02-02T19:23:50Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2024-02-02T19:26:01Z",
    "eventSource": "cloudfront.amazonaws.com",
    "eventName": "UpdateDistribution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "52.94.133.137",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/121.0.0.0 Safari/537.36",
    "requestParameters": {
        "distributionConfig": {
            "defaultRootObject": "",
            "aliases": {
                "quantity": 3,
                "items": [
                    "alejandro_rosalez.awsps.myinstance.com",
                    "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                    "*.alejandro_rosalez.awsps.myinstance.com"
                ]
            },
            "cacheBehaviors": {
                "quantity": 0,
                "items": []
            },
            "httpVersion": "http2and3",
            "originGroups": {
                "quantity": 0,
                "items": []
            },
            "viewerCertificate": {
                "minimumProtocolVersion": "TLSv1.2_2021",
                "cloudFrontDefaultCertificate": false,
                "aCMCertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                "sSLSupportMethod": "sni-only"
            },
            "webACLId": "arn:aws:wafv2:us-east-1:111122223333:global/webacl/testing-
acl/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "customErrorResponses": {
                "quantity": 0,
                "items": []
```

```
            },
            "logging": {
                "includeCookies": false,
                "prefix": "",
                "enabled": false,
                "bucket": ""
            },
            "priceClass": "PriceClass_All",
            "restrictions": {
                "geoRestriction": {
                    "restrictionType": "none",
                    "quantity": 0,
                    "items": []
                }
            },
            "isIPV6Enabled": true,
            "callerReference": "1578329170895",
            "continuousDeploymentPolicyId": "",
            "enabled": true,
            "defaultCacheBehavior": {
                "targetOriginId": "d111111abcdef8",
                "minTTL": 0,
                "compress": false,
                "maxTTL": 31536000,
                "functionAssociations": {
                    "quantity": 0,
                    "items": []
                },
                "trustedKeyGroups": {
                    "quantity": 0,
                    "items": [],
                    "enabled": false
                },
                "smoothStreaming": false,
                "fieldLevelEncryptionId": "",
                "defaultTTL": 86400,
                "lambdaFunctionAssociations": {
                    "quantity": 0,
                    "items": []
                },
                "viewerProtocolPolicy": "redirect-to-https",
                "forwardedValues": {
                    "cookies": {"forward": "none"},
                    "queryStringCacheKeys": {
```

```
                    "quantity": 0,
                    "items": []
                },
                "queryString": false,
                "headers": {
                    "quantity": 1,
                    "items": ["*"]
                }
            },
            "trustedSigners": {
                "items": [],
                "enabled": false,
                "quantity": 0
            },
            "allowedMethods": {
                "quantity": 2,
                "items": [
                    "HEAD",
                    "GET"
                ],
                "cachedMethods": {
                    "quantity": 2,
                    "items": [
                        "HEAD",
                        "GET"
                    ]
                }
            }
        },
        "staging": false,
        "origins": {
            "quantity": 1,
            "items": [
                {
                    "originPath": "",
                    "connectionTimeout": 10,
                    "customOriginConfig": {
                        "originReadTimeout": 30,
                        "hTTPSPort": 443,
                        "originProtocolPolicy": "https-only",
                        "originKeepaliveTimeout": 5,
                        "hTTPPort": 80,
                        "originSslProtocols": {
                            "quantity": 3,
```

```
                            "items": [
                                "TLSv1",
                                "TLSv1.1",
                                "TLSv1.2"
                            ]
                        }
                    },
                    "id": "d111111abcdef8",
                    "domainName": "d111111abcdef8.cloudfront.net",
                    "connectionAttempts": 3,
                    "customHeaders": {
                        "quantity": 0,
                        "items": []
                    },
                    "originShield": {"enabled": false},
                    "originAccessControlId": ""
                }
            ]
        },
        "comment": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    "id": "EDFDVBD6EXAMPLE",
    "ifMatch": "E1RTLUR9YES76O"
},
"responseElements": {
    "distribution": {
        "activeTrustedSigners": {
            "quantity": 0,
            "enabled": false
        },
        "id": "EDFDVBD6EXAMPLE",
        "domainName": "d111111abcdef8.cloudfront.net",
        "distributionConfig": {
            "defaultRootObject": "",
            "aliases": {
                "quantity": 3,
                "items": [
                    "alejandro_rosalez.awsps.myinstance.com",
                    "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                    "*.alejandro_rosalez.awsps.myinstance.com"
                ]
            },
            "cacheBehaviors": {"quantity": 0},
            "httpVersion": "http2and3",
```

```
                "originGroups": {"quantity": 0},
                "viewerCertificate": {
                    "minimumProtocolVersion": "TLSv1.2_2021",
                    "cloudFrontDefaultCertificate": false,
                    "aCMCertificateArn": "arn:aws:acm:us-
east-1:111122223333:certificate/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
                    "sSLSupportMethod": "sni-only",
                    "certificateSource": "acm",
                    "certificate": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
                },
                "webACLId": "arn:aws:wafv2:us-east-1:111122223333:global/webacl/
testing-acl/a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
                "customErrorResponses": {"quantity": 0},
                "logging": {
                    "includeCookies": false,
                    "prefix": "",
                    "enabled": false,
                    "bucket": ""
                },
                "priceClass": "PriceClass_All",
                "restrictions": {
                    "geoRestriction": {
                        "restrictionType": "none",
                        "quantity": 0
                    }
                },
                "isIPV6Enabled": true,
                "callerReference": "1578329170895",
                "continuousDeploymentPolicyId": "",
                "enabled": true,
                "defaultCacheBehavior": {
                    "targetOriginId": "d111111abcdef8",
                    "minTTL": 0,
                    "compress": false,
                    "maxTTL": 31536000,
                    "functionAssociations": {"quantity": 0},
                    "trustedKeyGroups": {
                        "quantity": 0,
                        "enabled": false
                    },
                    "smoothStreaming": false,
                    "fieldLevelEncryptionId": "",
                    "defaultTTL": 86400,
```

```
                    "lambdaFunctionAssociations": {"quantity": 0},
                    "viewerProtocolPolicy": "redirect-to-https",
                    "forwardedValues": {
                        "cookies": {"forward": "none"},
                        "queryStringCacheKeys": {"quantity": 0},
                        "queryString": false,
                        "headers": {
                            "quantity": 1,
                            "items": ["*"]
                        }
                    },
                    "trustedSigners": {
                        "enabled": false,
                        "quantity": 0
                    },
                    "allowedMethods": {
                        "quantity": 2,
                        "items": [
                            "HEAD",
                            "GET"
                        ],
                        "cachedMethods": {
                            "quantity": 2,
                            "items": [
                                "HEAD",
                                "GET"
                            ]
                        }
                    }
                },
                "staging": false,
                "origins": {
                    "quantity": 1,
                    "items": [
                        {
                            "originPath": "",
                            "connectionTimeout": 10,
                            "customOriginConfig": {
                                "originReadTimeout": 30,
                                "hTTPSPort": 443,
                                "originProtocolPolicy": "https-only",
                                "originKeepaliveTimeout": 5,
                                "hTTPPort": 80,
                                "originSslProtocols": {
```

```
                                "quantity": 3,
                                "items": [
                                    "TLSv1",
                                    "TLSv1.1",
                                    "TLSv1.2"
                                ]
                            }
                        },
                        "id": "d111111abcdef8",
                        "domainName": "d111111abcdef8.cloudfront.net",
                        "connectionAttempts": 3,
                        "customHeaders": {"quantity": 0},
                        "originShield": {"enabled": false},
                        "originAccessControlId": ""
                    }
                ]
            },
            "comment": "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "aliasICPRecordals": [
            {
                "cNAME": "alejandro_rosalez.awsps.myinstance.com",
                "iCPRecordalStatus": "APPROVED"
            },
            {
                "cNAME": "cross-testing.alejandro_rosalez.awsps.myinstance.com",
                "iCPRecordalStatus": "APPROVED"
            },
            {
                "cNAME": "*.alejandro_rosalez.awsps.myinstance.com",
                "iCPRecordalStatus": "APPROVED"
            }
        ],
        "aRN": "arn:aws:cloudfront::111122223333:distribution/EDFDVBD6EXAMPLE",
        "status": "InProgress",
        "lastModifiedTime": "Feb 2, 2024 7:26:01 PM",
        "activeTrustedKeyGroups": {
            "enabled": false,
            "quantity": 0
        },
        "inProgressInvalidationBatches": 0
    },
    "eTag": "E1YHBLAB2BJY1G"
},
```

```
    "requestID": "4e6b66f9-d548-11e3-a8a9-73e33example",
    "eventID": "5ab02562-0fc5-43d0-b7b6-90293example",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "apiVersion": "2020_05_31",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "cloudfront.amazonaws.com"
    },
    "sessionCredentialFromConsole": "true"
}
```

例如：UpdateKeys

下面的示例显示了一个 CloudTrail 事件，该事件说明了 UpdateKeys 操作。

对于对 CloudFront KeyValueStore API 的调用，`eventSource` 是 `edgekeyvaluestore.amazonaws.com`，而不是 `cloudfront.amazonaws.com`。

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:role-session-name",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/role-session-name",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2023-11-01T23:41:14Z",
                "mfaAuthenticated": "false"
            }
```

```
        }
    },
    "eventTime": "2023-11-01T23:41:28Z",
    "eventSource": "edgekeyvaluestore.amazonaws.com",
    "eventName": "UpdateKeys",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "3.235.183.252",
    "userAgent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/121.0.0.0 Safari/537.36,
    "requestParameters": {
        "kvsARN": "arn:aws:cloudfront::111122223333:key-value-store/a1b2c3d4-5678-90ab-
cdef-EXAMPLE11111",
        "ifMatch": "KV3O6B1CX531EBP",
        "deletes": [
            {"key": "key1"}
        ]
    },
    "responseElements": {
        "itemCount": 0,
        "totalSizeInBytes": 0,
        "eTag": "KVDC9VEVZ71ZGO"
    },
    "requestID": "5ccf104c-acce-4ea1-b7fc-73e33example",
    "eventID": "a0b1b5c7-906c-439d-9925-90293example",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::CloudFront::KeyValueStore",
            "ARN": "arn:aws:cloudfront::111122223333:key-value-store/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "111122223333.cloudfront-kvs.global.api.aws"
    }
}
```

有关 CloudTrail 记录内容的信息，请参阅《AWS CloudTrail 用户指南》中的 CloudTrail 记录内容。

使用 AWS Config 跟踪配置更改

使用 AWS Config 可记录对 CloudFront 分配设置进行的配置更改。您可以捕获对分配状态、价格级别、源、地理限制设置和 Lambda@Edge 配置进行的更改。

> **Note**
>
> AWS Config 不记录 CloudFront 流分配的键/值标签。

使用 CloudFront 设置 AWS Config

在设置 AWS Config 时，您可以选择记录所有受支持的 AWS 资源，也可以只记录某些指定资源（例如，仅记录对 CloudFront 的更改）。要查看受支持 CloudFront 资源的列表，请参阅《AWS Config 开发人员指南》中“受支持的资源类型”主题的 Amazon CloudFront 部分。

要跟踪对 CloudFront 分配进行的配置更改，您必须登录美国东部（弗吉尼亚州北部）AWS 区域的 CloudFront 控制台。

> **Note**
>
> 使用 AWS Config 记录资源可能会有延迟。AWS Config 仅在发现资源后记录资源。

Console

使用 CloudFront设置 AWS Config（控制台）

1. 登录到 AWS Management Console，然后通过以下网址打开 AWS Config 控制台：https://console.aws.amazon.com/config/。
2. 选择 Get Started Now。
3. 在设置页面上，为要记录的资源类型指定您希望 AWS 记录的 AWS Config 资源类型。如果您希望仅记录 CloudFront 更改，请选择特定类型，然后在 CloudFront 下面选择要跟踪更改的分配或流分配。

   要添加或更改要跟踪的分配，请在完成初始设置后选择左侧的设置。

4. 为 AWS Config指定额外的必需选项：设置一个通知，为配置信息指定一个位置，然后添加用于评估资源类型的规则。

有关更多信息，请参阅《AWS Config 开发人员指南》中的使用控制台设置 AWS Config。

AWS CLI

要通过 CloudFront 使用 AWS CLI 设置 AWS Config，请参阅《AWS Config 开发人员指南》中的使用 AWS CLI 设置 AWS Config。

AWS Config API

要通过 CloudFront 使用 AWS Config API 设置 AWS Config，请参阅《AWS Config API 参考》中的 StartConfigurationRecorder 操作和其他信息。

查看 CloudFront 配置历史记录

在 AWS Config 开始记录您的分配的配置更改后，您可以获取为 CloudFront 配置的任何分配的配置历史记录。

您可以使用以下方式查看配置历史记录。

Console

对于每个已记录的资源，您可以查看时间线页面，该页面提供了配置详细信息的历史记录。要查看此页面，请选择专用主机页面的配置时间线列中的灰色图标。

有关更多信息，请参阅《AWS Config 开发人员指南》 中的在 AWS Config 控制台中查看配置详细信息。

AWS CLI

要获取所有分配的列表，请运行 list-discovered-resources 命令，如以下示例所示。

```
aws configservice list-discovered-resources --resource-type
 AWS::CloudFront::Distribution
```

要获取特定时间间隔内某个分配的配置详细信息，请运行 get-resource-config-history 命令。

有关更多信息，请参阅《AWS Config 开发人员指南》 中的使用 CLI 查看配置详细信息。

AWS Config API

要获取所有分配的列表，请执行 ListDiscoveredResources 操作。

要获取特定时间间隔内某个分配的配置详细信息，请使用 GetResourceConfigHistory 操作。有关详细信息，请参阅 AWS Config API 参考。

Amazon CloudFront 中的安全性

AWS 十分重视云安全性。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。作为 AWS 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于 Amazon CloudFront 的合规性计划，请参阅合规性计划范围内内的AWS服务。
- 云中的安全性 - 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、您组织的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 CloudFront 时应用责任共担模式。以下主题说明了如何配置 CloudFront 以实现您的安全性和合规性目标。您还将了解如何使用其他AWS服务来帮助您监控和保护您的 CloudFront 资源。

主题

Amazon CloudFront 中的数据保护

AWS 责任共担模式适用于 Amazon CloudFront 中的数据保护。如该模式中所述，AWS 负责保护运行所有 AWS Cloud 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 AWS 责任共担模式和 GDPR 博客文章。

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）设置单个用户。这样，每个用户只获得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 AWS 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用 AWS CloudTrail 设置 API 和用户活动日记账记录。
- 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 中的敏感数据。
- 如果在通过命令行界面或 API 访问 AWS 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您通过控制台、API、AWS CLI 或 AWS SDK 使用 CloudFront 或其他 AWS 服务时。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

Amazon CloudFront 提供了几个选项，您可使用这些选项来帮助保护所提供的内容：

- 配置 HTTPS 连接。
- 配置字段级加密以在传输过程中为指定数据提供额外的安全性。
- 限制对内容的访问权限，从而只有特定人员或者特定区域中的人员可以查看这些内容。

以下主题更详细地说明了选项。

主题

传输中加密

要对传输中的数据加密，您可以将 Amazon CloudFront 配置为要求查看器使用 HTTPS 请求您的文件，以便在 CloudFront 与查看器通信时加密连接。您也可以将 CloudFront 配置为使用 HTTPS 从源获取文件，以便在 CloudFront 与源通信时加密连接。

有关更多信息，请参阅 将 HTTPS 与 CloudFront 结合使用。

字段级加密增加了一个额外的安全保护层以及 HTTPS，使您可以在整个系统处理过程中保护特定的数据，以便只有某些应用程序才能查看它。通过在 CloudFront 中配置字段级加密，您可以将用户提交的敏感信息安全上传到您的 Web 服务器。客户端提供的敏感信息在更接近用户的边缘站点中进行加密。这些敏感信息在整个应用程序堆栈中保持加密状态，从而确保仅需要该数据（以及具有用于解密的凭证）的应用程序才能使用该数据。

有关更多信息，请参阅 使用字段级加密帮助保护敏感数据。

CloudFront API 终端节点 `cloudfront.amazonaws.com` 和 `cloudfront-fips.amazonaws.com` 只接受 HTTPS 流量。这意味着，当您使用 CloudFront API 发送和接收信息时，您的数据（包括分配配置、缓存策略和源请求策略、密钥组和公钥以及 CloudFront Functions 中的函数代码）始终在传输过程中加密。此外，发送到 CloudFront API 终端节点的所有请求都使用 AWS 凭证进行签名并记录到 AWS CloudTrail 中。

CloudFront Functions 中的函数代码和配置被复制到边缘站点节点 (POP) 以及在 CloudFront 使用的其他存储位置之间复制时，始终在传输过程中加密。

静态加密

CloudFront Functions 中的函数代码和配置始终以加密的格式存储在边缘站点 POP 上和 CloudFront 使用的其他存储位置中。

限制对内容的访问

许多通过互联网分发内容的公司都希望限制对文档、业务数据、流媒体或面向一部分用户的内容的访问。要使用 Amazon CloudFront 安全地提供此内容，您可执行以下一个或多个操作：

使用签名 URL 或 Cookie

> 您通过 CloudFront，使用签名 URL 或签名 Cookie 来提供此私有内容，从而限制面向选定用户（例如，付费用户）的内容的访问。有关更多信息，请参阅 使用签名 URL 和签名 Cookie 提供私有内容。

限制对 Amazon S3 存储桶中内容的访问

> 例如，如果您通过使用 CloudFront 签名 URL 或签名 Cookie 来限制对内容的访问，您也不希望人们使用文件的直接 URL 查看文件。而是希望他们只能通过使用 CloudFront URL 访问文件，这样您的保护才会发挥作用。
>
> 如果您使用 Amazon S3 存储桶作为 CloudFront 分配的源，则可以设置源访问控制 (OAC) 来限制对 S3 存储桶的访问。有关更多信息，请参阅 the section called “限制对 Amazon Simple Storage Service 源的访问”。

限制对 Application Load Balancer 提供的内容的访问

当您将 CloudFront 与 Elastic Load Balancing 中的 Application Load Balancer 作为源时，您可以配置 CloudFront 以防止用户直接访问 Application Load Balancer。这使得用户只能通过 CloudFront 访问 Application Load Balancer，从而确保您获得使用 CloudFront 的益处。有关更多信息，请参阅限制访问应用程序负载均衡器。

使用 AWS WAF Web ACL

您可以使用 Web 应用程序防火墙服务 AWS WAF 创建 Web 访问控制列表 (Web ACL) 来限制对内容的访问。根据指定的条件（如请求源自的 IP 地址或查询字符串的值），CloudFront 会使用所请求的内容或使用 HTTP 状态代码 403（禁止）来响应请求。有关更多信息，请参阅 使用 AWS WAF 保护功能。

使用地理限制

您可以使用地理限制（也称为地理阻止）禁止特定地理位置的用户访问您通过 CloudFront 分配提供的内容。配置地理限制时有多个选项可供选择。有关更多信息，请参阅 限制您的内容的地理分配。

适用于 Amazon CloudFront 的 Identity and Access Management

AWS Identity and Access Management（IAM）是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制可以通过身份验证（登录）和授权（具有权限）使用 CloudFront 资源的人员。IAM 是一项无需额外费用即可使用的 AWS 服务。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- Amazon CloudFront 如何与 IAM 结合工作
- 适用于 Amazon CloudFront 的基于身份的策略示例
- 适用于 Amazon CloudFront 的 AWS 托管策略
- 排查 Amazon CloudFront 身份和访问问题

受众

使用 AWS Identity and Access Management（IAM）的方式因您可以在 CloudFront 中执行的操作而异。

服务用户 – 如果您使用 CloudFront 服务来完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多 CloudFront 功能来完成工作时，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 CloudFront 中的某项功能，请参阅排查 Amazon CloudFront 身份和访问问题。

服务管理员 - 如果您在公司负责管理 CloudFront 资源，则您可能具有 CloudFront 的完全访问权限。您有责任确定您的服务用户应访问哪些 CloudFront 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 CloudFront 结合使用的更多信息，请参阅Amazon CloudFront 如何与 IAM 结合工作。

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 CloudFront 的访问的详细信息。要查看您可在 IAM 中使用的 CloudFront 基于身份的策略示例，请参阅适用于 Amazon CloudFront 的基于身份的策略示例。

使用身份进行身份验证

身份验证是您使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过代入 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center（IAM Identity Center）用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。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。当您使用联合身份验证访问 AWS 时，您就是在间接代入角色。

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

如果您以编程方式访问 AWS，AWS 将提供软件开发工具包 (SDK) 和命令行界面 (CLI)，以便使用您的凭证以加密方式签署您的请求。如果您不使用 AWS 工具，则必须自行对请求签名。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 根用户

当您创建 AWS 账户 时，最初使用的是一个对账户中所有 AWS 服务 和资源拥有完全访问权限的登录身份。此身份称为 AWS 账户 根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。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）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 AWS 服务。

联合身份是来自企业用户目录、Web 身份提供程序、AWS Directory Service、Identity Center 目录的用户，或任何使用通过身份源提供的凭证来访问 AWS 服务 的用户。当联合身份访问 AWS 账户 时，他们代入角色，而角色提供临时凭证。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和组，也可以连接并同步到您自己的身份源中的一组用户和组以跨所有 AWS 账户 和应用程序使用。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

IAM 用户是 AWS 账户 内对某个人员或应用程序具有特定权限的一个身份。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，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

IAM 角色是 AWS 账户 中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时代入 IAM 角色。您可以调用 AWS CLI 或 AWS

API 操作或使用自定义网址以担任角色。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：

- 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 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 – 某些 AWS 服务 使用其他 AWS 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 EC2 中运行应用程序或在 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：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在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 – 服务相关角色是与 AWS 服务 关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon 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凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到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

您将创建策略并将其附加到 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。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内容。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 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，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。具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 策略。

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。托管式策略是可以附加到 AWS 账户 中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户管理型策略。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。主体可以包括账户、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来自 IAM 的 AWS 托管式策略。

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。

Simple Storage Service（Amazon S3）、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。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。

其他策略类型

AWS 支持额外的、不太常用的策略类型。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功能，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，指定了组织或组织单位（OU）在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）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 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅《IAM 用户指南》中的策略评估逻辑。

Amazon CloudFront 如何与 IAM 结合工作

在使用 IAM 管理对 CloudFront 的访问之前，您应该了解哪些 IAM 功能可用于 CloudFront。

可以与 Amazon CloudFront 结合使用的 IAM 功能

IAM 功能	CloudFront 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（特定于服务）	是
ACL	否
ABAC（策略中的标签）	部分
临时凭证	是
转发访问会话 (FAS)	否
服务角色	否
服务相关角色	是

要大致了解 CloudFront 和其他 AWS 服务如何与大多数 IAM 功能结合使用，请参阅《IAM 用户指南》中的与 IAM 结合使用的 AWS 服务。

适用于 CloudFront 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 策略。

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 CloudFront 的基于身份的策略示例

要查看 CloudFront 基于身份的策略的示例，请参阅适用于 Amazon CloudFront 的基于身份的策略示例。

CloudFront 内基于资源的策略

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。主体可以包括账户、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他账户中的 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。当主体和资源处于不同的AWS账户中时，则信任账户中的 IAM 管理员还必须授予主体实体（用户或角色）对资源的访问权限。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。

CloudFront 的策略操作

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。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。

要查看 CloudFront 操作的列表，请参阅《服务授权参考》中的 Amazon CloudFront定义的操作。

CloudFront 中的策略操作在操作前使用以下前缀：

```
cloudfront
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "cloudfront:action1",
      "cloudfront:action2"
         ]
```

要查看 CloudFront 基于身份的策略的示例，请参阅适用于 Amazon CloudFront 的基于身份的策略示例。

CloudFront 的策略资源

支持策略资源	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。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 CloudFront 资源类型及其 ARN 的列表，请参阅《服务授权参考》中的 Amazon CloudFront 定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 Amazon CloudFront 定义的操作。

要查看 CloudFront 基于身份的策略的示例，请参阅适用于 Amazon CloudFront 的基于身份的策略示例。

CloudFront 的策略条件键

支持特定于服务的策略条件键	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。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 AWS 使用逻辑 `OR` 运算来评估条件。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 策略元素：变量和标签。

AWS支持全局条件键和特定于服务的条件键。要查看所有 AWS 全局条件键，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

有关 CloudFront 条件键的列表，请参阅《服务授权参考》中的 Amazon CloudFront 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon CloudFront 定义的操作。

要查看 CloudFront 基于身份的策略的示例，请参阅适用于 Amazon CloudFront 的基于身份的策略示例。

CloudFront 中的 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。

ABAC 以及 CloudFront

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 AWS 中，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）以及AWS资源。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，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 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）。

CloudFront 仅支持使用 ABAC 进行分配。

将临时凭证用于 CloudFront

支持临时凭证	是

某些 AWS 服务在您使用临时凭证登录时无法正常工作。有关更多信息，包括 AWS 服务与临时凭证配合使用，请参阅《IAM 用户指南》中的使用 IAM 的 AWS 服务。

如果您不使用用户名和密码而用其他方法登录到AWS Management Console，则使用临时凭证。例如，当您使用贵公司的单点登录（SSO）链接访问AWS时，该过程将自动创建临时凭证。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 或者 AWS API 创建临时凭证。之后，您可以使用这些临时凭证访问AWS。AWS 建议您动态生成临时凭证，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。

CloudFront 的转发访问会话

支持转发访问会话（FAS）	否

当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在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。

CloudFront 的服务角色

支持服务角色	否

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破坏 CloudFront 的功能。仅当 CloudFront 提供相关指导时才编辑服务角色。

CloudFront 的服务相关角色

支持服务相关角色	是

服务相关角色是一种与 AWS 服务 相关的服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 AWS 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

Lambda@Edge 使用服务相关角色为您执行操作。有关创建或管理 CloudFront 服务相关角色的更多信息，请参阅Lambda@Edge 的服务相关角色。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的 AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

适用于 Amazon CloudFront 的基于身份的策略示例

默认情况下，用户和角色没有创建或修改 CloudFront 资源的权限。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 策略。管理员随后可以向角色添加 IAM 策略，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 策略。

有关 CloudFront 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 Amazon CloudFront 的操作、资源和条件键。

主题

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 CloudFront 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：

- AWS 托管策略及转向最低权限许可入门 – 要开始向用户和工作负载授予权限，请使用 AWS 托管策略来为许多常见使用场景授予权限。您可以在 AWS 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 AWS 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略或工作职能的 AWS 托管策略。
- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。

- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定（AWS 服务例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证（MFA）– 如果您所处的场景要求您的 AWS 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。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 CloudFront 控制台

要访问 Amazon CloudFront 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 AWS 账户中的 CloudFront 资源的详细信息。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。

对于只需要调用 AWS CLI 或 AWS API 的用户，无需为其提供最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍可使用 CloudFront 控制台，请同时将 CloudFront *ConsoleAccess* 或 *ReadOnly* AWS 托管策略添加到实体。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完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
    "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": "*"
        }
    ]
}
```

以编程方式访问 CloudFront 的权限

下面显示了一个权限策略。Sid 或语句 ID 是可选的。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "AllowAllCloudFrontPermissions",
         "Effect": "Allow",
         "Action": ["cloudfront:*"],
         "Resource": "*"
      }
   ]
}
```

该策略授予执行所有 CloudFront 操作的权限，这足以通过编程方式访问 CloudFront。如果您使用控制台访问 CloudFront，请参阅使用 CloudFront 控制台所需的权限。

有关您为授予或拒绝使用每项操作的权限而指定的操作和 ARN 的列表，请参阅《服务授权参考》中的 Amazon CloudFront 的操作、资源和条件键。

使用 CloudFront 控制台所需的权限

要授予对 CloudFront 控制台的完全访问权，您可以在以下权限策略中授予权限：

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:*",
            "cloudwatch:DescribeAlarms",
            "cloudwatch:PutMetricAlarm",
            "cloudwatch:GetMetricStatistics",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets",
            "s3:PutBucketPolicy"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

下面是需要权限的原因：

acm:ListCertificates

如果您使用 CloudFront 控制台创建和更新分配，并且希望配置 CloudFront 以要求在查看器与 CloudFront 之间或 CloudFront 与源之间使用 HTTPS，则可查看 ACM 证书的列表。

如果您未使用 CloudFront 控制台，则不需要此权限。

cloudfront:*

允许您执行所有 CloudFront 操作。

cloudwatch:DescribeAlarms 和 **cloudwatch:PutMetricAlarm**

允许您在 CloudFront 控制台中创建和查看 CloudWatch 警报。另请参阅 `sns:ListSubscriptionsByTopic` 和 `sns:ListTopics`。

如果您未使用 CloudFront 控制台，则不需要这些权限。

cloudwatch:GetMetricStatistics

允许 CloudFront 在 CloudFront 控制台中呈现 CloudWatch 指标。

如果您未使用 CloudFront 控制台，则不需要此权限。

elasticloadbalancing:DescribeLoadBalancers

在创建和更新分配时，您可以查看可用源列表中的 Elastic Load Balancing 负载均衡器的列表。

如果您未使用 CloudFront 控制台，则不需要此权限。

iam:ListServerCertificates

如果您使用 CloudFront 控制台创建和更新分配，并且希望配置 CloudFront 以要求在查看器与 CloudFront 之间或 CloudFront 与源之间使用 HTTPS，则可查看 IAM 证书存储中的证书列表。

如果您未使用 CloudFront 控制台，则不需要此权限。

s3:ListAllMyBuckets

当您创建和更新分配时，您可以执行以下操作：

- 在可用源列表中查看 S3 存储桶的列表
- 查看可将访问日志保存到的 S3 存储桶的列表

如果您未使用 CloudFront 控制台，则不需要此权限。

`S3:PutBucketPolicy`

当您创建或更新将限制对 S3 存储桶的访问的分配时，用户可以更新存储桶策略以授予对 CloudFront 源访问身份的访问权。有关更多信息，请参阅 the section called “使用源访问身份（旧版，不推荐）”。

如果您未使用 CloudFront 控制台，则不需要此权限。

`sns:ListSubscriptionsByTopic` 和 **`sns:ListTopics`**

当您在 CloudFront 控制台中创建 CloudWatch 警报时，可以为通知选择 SNS 主题。

如果您未使用 CloudFront 控制台，则不需要这些权限。

`waf:GetWebACL` 和 **`waf:ListWebACLs`**

允许您在 CloudFront 控制台中查看 AWS WAF Web ACL 的列表。

如果您未使用 CloudFront 控制台，则不需要这些权限。

用于 CloudFront 的 AWS 托管（预定义）策略

AWS通过提供由创建和管理的独立 IAM 策略来满足许多常用案例的要求。AWS这些 AWS 托管策略可针对常用案例授予必要的权限，使您免去调查所需权限的工作。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略。对于 CloudFront，IAM 提供了两个托管策略：

- CloudFrontFullAccess – 授予对 CloudFront 资源的完全访问权限。

> **Important**
>
> 如果您希望 CloudFront 创建和保存访问日志，则需要授予其他权限。有关更多信息，请参阅配置标准日志记录和访问您的日志文件所需的权限。

- CloudFrontReadOnlyAccess – 授予对 CloudFront 资源的只读访问权限。

客户管理的策略示例

您可以创建自己的自定义 IAM 策略，以授予执行 CloudFront API 操作的相关权限。您可以将这些自定义策略附加到需要指定权限的 IAM 用户或组。当您使用 CloudFront API、AWS 开发工具包或 AWS CLI 时，可以使用这些策略。以下示例显示了几个常见使用案例的权限。有关向用户授予对 CloudFront 的完全访问权限的策略，请参阅使用 CloudFront 控制台所需的权限。

示例

示例 1：允许对所有分配进行读访问

以下权限策略向用户授予在 CloudFront 控制台中查看所有分配的权限：

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:GetDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

示例 2：允许创建、更新和删除分配

以下权限策略允许用户使用 CloudFront 控制台来创建、更新和删除分配：

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:CreateDistribution",
            "cloudfront:DeleteDistribution",
            "cloudfront:GetDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:UpdateDistribution",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets",
            "s3:PutBucketPolicy"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

`cloudfront:ListCloudFrontOriginAccessIdentities` 权限允许用户自动向现有的源访问身份授予对 Amazon S3 存储桶中的对象的访问权。如果您还希望用户能够创建源访问身份，则还需要授予 `cloudfront:CreateCloudFrontOriginAccessIdentity` 权限。

示例 3：允许创建和列出失效

以下权限策略允许用户创建和列出失效。它包括对 CloudFront 分配的读访问权，因为您通过先显示分配的设置来创建和查看失效：

```
{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "acm:ListCertificates",
            "cloudfront:GetDistribution",
            "cloudfront:GetStreamingDistribution",
            "cloudfront:GetDistributionConfig",
            "cloudfront:ListDistributions",
            "cloudfront:ListCloudFrontOriginAccessIdentities",
            "cloudfront:CreateInvalidation",
            "cloudfront:GetInvalidation",
            "cloudfront:ListInvalidations",
            "elasticloadbalancing:DescribeLoadBalancers",
            "iam:ListServerCertificates",
            "sns:ListSubscriptionsByTopic",
            "sns:ListTopics",
            "waf:GetWebACL",
            "waf:ListWebACLs"
         ],
         "Resource":"*"
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListAllMyBuckets"
         ],
         "Resource":"arn:aws:s3:::*"
      }
   ]
}
```

示例 4：允许创建分配

以下权限策略向用户授予在 CloudFront 控制台中创建和列出分配的权限：要执行 `CreateDistribution` 操作，请为 `Resource` 指定通配符（`*`），而不是为分配

ARN（`arn:aws:cloudfront::123456789012:distribution/*`）指定通配符。有关 `Resource` 元素的更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：资源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "cloudfront:CreateDistribution",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "cloudfront:ListDistributions",
            "Resource": "*"
        }
    ]
}
```

适用于 Amazon CloudFront 的 AWS 托管策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 AWS 托管策略更简单。仅为用户提供所需权限来创建 IAM 客户托管策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管策略。这些策略涵盖常见使用案例，可在您的 AWS 账户 中使用。有关 AWS 托管策略的更多信息，请参阅《IAM 用户指南》中的AWS 托管策略。

AWS 服务负责维护和更新 AWS 托管策略。您无法更改 AWS 托管策略中的权限。服务偶尔会向 AWS 托管策略添加额外权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当新功能启动或新权限可用时，服务最有可能会更新 AWS 托管策略。服务不会从 AWS 托管策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，AWS还支持跨多种服务的工作职能的托管策略。例如，ReadOnlyAccess AWS 托管式策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动新功能时，AWS 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅《IAM 用户指南》中的适用于工作职能的AWS托管策略。

AWS托管策略：CloudFrontReadOnlyAccess

您可以将 CloudFrontReadOnlyAccess 策略附加到 IAM 身份。此策略允许 CloudFront 资源的只读权限。它还允许与 CloudFront 相关且在 CloudFront 控制台中可见的其它 AWS 服务资源的只读权限。

权限详细信息

该策略包含以下权限。

- `cloudfront:Describe*` – 允许委托人获取有关 CloudFront 资源的元数据信息。
- `cloudfront:Get*` – 允许委托人获取 CloudFront 资源的详细信息和配置。
- `cloudfront:List*` – 允许委托人获取 CloudFront 资源列表。
- `cloudfront-keyvaluestore:Describe*` - 允许委托人获取有关键值存储的信息。
- `cloudfront-keyvaluestore:Get*` - 允许委托人获取键值存储的详细信息和配置。
- `cloudfront-keyvaluestore:List*` - 允许委托人获取键值存储的列表。
- `acm:ListCertificates` – 允许委托人获取 ACM 证书列表。
- `iam:ListServerCertificates` – 允许委托人获取存储在 IAM 中的服务器证书列表。
- `route53:List*` – 允许委托人获取 Route 53 资源列表。
- `waf:ListWebACLs` – 允许委托人在 AWS WAF 中获取 Web ACL 列表。
- `waf:GetWebACL` – 允许委托人在 AWS WAF 中获取有关 Web ACL 的详细信息。
- `wafv2:ListWebACLs` – 允许委托人在 AWS WAF 中获取 Web ACL 列表。
- `wafv2:GetWebACL` – 允许委托人在 AWS WAF 中获取有关 Web ACL 的详细信息。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "cfReadOnly",
      "Effect": "Allow",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:Describe*",
        "cloudfront:Get*",
        "cloudfront:List*",
        "cloudfront-keyvaluestore:Describe*",
        "cloudfront-keyvaluestore:Get*",
        "cloudfront-keyvaluestore:List*",
```

```
        "iam:ListServerCertificates",
        "route53:List*",
        "waf:ListWebACLs",
        "waf:GetWebACL",
        "wafv2:ListWebACLs",
        "wafv2:GetWebACL"
      ],
      "Resource": "*"
    }
  ]
}
```

AWS托管策略：CloudFrontFullAccess

您可以将 CloudFrontFullAccess 策略附加到 IAM 身份。此策略允许 CloudFront 资源的管理权限。它还允许与 CloudFront 相关且在 CloudFront 控制台中可见的其它 AWS 服务资源的只读权限。

权限详细信息

该策略包含以下权限。

- `s3:ListAllMyBuckets` – 允许委托人获取所有 Amazon S3 存储桶的列表。
- `acm:ListCertificates` – 允许委托人获取 ACM 证书列表。
- `cloudfront:*` – 允许委托人对所有 CloudFront 资源执行所有操作。
- `cloudfront-keyvaluestore:*` - 允许委托人对键值存储执行所有操作。
- `iam:ListServerCertificates` – 允许委托人获取存储在 IAM 中的服务器证书列表。
- `waf:ListWebACLs` – 允许委托人在 AWS WAF 中获取 Web ACL 列表。
- `waf:GetWebACL` – 允许委托人在 AWS WAF 中获取有关 Web ACL 的详细信息。
- `wafv2:ListWebACLs` – 允许委托人在 AWS WAF 中获取 Web ACL 列表。
- `wafv2:GetWebACL` – 允许委托人在 AWS WAF 中获取有关 Web ACL 的详细信息。
- `kinesis:ListStreams` – 允许委托人获取 Amazon Kinesis 流的列表。
- `kinesis:DescribeStream` – 允许委托人获取有关 Kinesis 流的详细信息。
- `iam:ListRoles` – 允许委托人在 IAM 中获取角色列表。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "cfflistbuckets",
      "Action": [
        "s3:ListAllMyBuckets"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "cffullaccess",
      "Action": [
        "acm:ListCertificates",
        "cloudfront:*",
        "cloudfront-keyvaluestore:*",
        "iam:ListServerCertificates",
        "waf:ListWebACLs",
        "waf:GetWebACL",
        "wafv2:ListWebACLs",
        "wafv2:GetWebACL",
        "kinesis:ListStreams"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Sid": "cffdescribestream",
      "Action": [
        "kinesis:DescribeStream"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:kinesis:*:*:*"
    },
    {
      "Sid": "cfflistroles",
      "Action": [
        "iam:ListRoles"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:iam::*:*"
    }
  ]
}
```

AWS 托管策略：AWSCloudFrontLogger

您不能将 AWSCloudFrontLogger 策略添加到您的 IAM 身份。此附加到服务相关角色的策略允许 CloudFront 代表您执行操作。有关更多信息，请参阅the section called “Lambda@Edge 的服务相关角色”。

此策略允许 CloudFront 将日志文件推送到 Amazon CloudWatch。有关此策略中包含的权限的详细信息，请参阅the section called “CloudFront Logger 的服务相关角色权限”。

AWS托管策略：AWSLambDAccessReplicator

您不能将 AWSLambdagReplicator 策略添加到 IAM 身份。此附加到服务相关角色的策略允许 CloudFront 代表您执行操作。有关更多信息，请参阅the section called “Lambda@Edge 的服务相关角色”。

此策略允许在 AWS Lambda 中 CloudFront 创建、删除和禁用函数并将 Lambda@Edge 函数复制到 AWS 区域。有关此策略中包含的权限的详细信息，请参阅the section called “Lambda Replicator 的服务相关角色权限”。

CloudFront 对 AWS 托管策略做出的更新

查看有关 CloudFront 的 AWS 托管策略更新的详细信息（从该服务开始跟踪这些更改开始）。有关此页面更改的提示，请订阅文档历史记录页面上的 CloudFront RSS 馈送。

更改	描述	日期
CloudFrontReadOnlyAccess和CloudFrontFullAccess – 更新两个现有策略。	CloudFront 为键值存储添加了新权限 新的权限允许用户获取有关键值存储的信息并对键值存储执行操作。	2023 年 12 月 19 日
CloudFrontReadOnlyAccess – 更新了现有策略	CloudFront 添加了一个新的权限来描述 CloudFront Functions。	2021 年 9 月 8 日

更改	描述	日期
	此权限允许用户、组或角色读取有关函数的信息和元数据，但不能读取函数代码。	
CloudFront 已开启跟踪更改	CloudFront 为其AWS托管策略开启了更改跟踪。	2021 年 9 月 8 日

排查 Amazon CloudFront 身份和访问问题

使用以下信息可帮助您诊断和修复在使用 CloudFront 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 CloudFront 中执行操作
- 我无权执行 iam:PassRole
- 我希望允许我的 AWS 账户以外的人访问我的 CloudFront 资源

我无权在 CloudFront 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 `cloudfront:`*`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:
 cloudfront: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 `cloudfront:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系 AWS 管理员。您的管理员是提供登录凭证的人。

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 CloudFront。

有些 AWS 服务 允许将现有角色传递到该服务，而不是创建新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。

当名为 marymajor 的 IAM 用户尝试使用控制台在 CloudFront 中执行操作时，会发生以下示例错误。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 AWS 管理员。您的管理员是提供登录凭证的人。

我希望允许我的 AWS 账户以外的人访问我的 CloudFront 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 CloudFront 是否支持这些功能，请参阅Amazon CloudFront 如何与 IAM 结合工作。
- 要了解如何为您拥有的 AWS 账户中的资源提供访问权限，请参阅《IAM 用户指南》中的为您拥有的另一个 AWS 账户中的 IAM 用户提供访问权限。
- 要了解如何为第三方AWS 账户提供您的资源的访问权限，请参阅《IAM 用户指南》中的为第三方拥有的AWS 账户提供访问权限。
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。

Amazon CloudFront 中的日志记录和监控

监控是保持 CloudFront 和 AWS 解决方案的可用性和性能的重要环节。您应该从 AWS 解决方案的所有部分收集监控数据，以便您可以更轻松地调试多点故障（如果发生）。AWS 提供了多种工具来监控您的 CloudFront 资源和活动并对潜在事件做出响应：

Amazon CloudWatch 警报

使用 CloudWatch 警报，可以观看单个指标在指定时间段内的变化。如果指标超过给定阈值，则会向 Amazon SNS 主题或 AWS Auto Scaling 策略发送通知。当指标处于特定状态时，CloudWatch 警报不会调用操作。而是必须在状态已改变并在指定的若干个时间段内保持不变后才调用。有关更多信息，请参阅 使用 Amazon CloudWatch 监控 CloudFront 指标。

AWS CloudTrail 日志

CloudTrail 提供了用户、角色或 AWS 服务在 CloudFront 中所执行 API 操作的记录。使用 CloudTrail 收集的信息，您可以确定向 CloudFront 发出了什么 API 请求、发出请求的 IP 地址、何人发出的请求、发出请求的时间以及其他详细信息。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon CloudFront API 调用。

CloudFront 标准日志和实时日志

CloudFront 日志提供有关对分配做出的请求的详细记录。这些日志对许多应用程序都很有用。例如，日志信息可能在安全和访问权限审核方面很有用。有关更多信息，请参阅 CloudFront 和边缘函数日志记录。

边缘函数日志

CloudFront Functions 和 Lambda@Edge 等边缘函数生成的日志将直接发送到 Amazon CloudWatch Logs，而不会由 CloudFront 存储在任何地方。CloudFront Functions 使用 AWS Identity and Access Management (IAM) 服务相关角色将客户生成的日志在您的账户中直接发送到 CloudWatch Logs。

CloudFront 控制台报告

CloudFront 控制台包含多种报告，如缓存统计报告、常用对象报告和最用引用报告。大多数 CloudFront 控制台报告都基于 CloudFront 访问日志中的数据，其中包含有关 CloudFront 接收的每个用户请求的详细信息。不过，您无需启用访问日志即可查看此类报告。有关更多信息，请参阅 在控制台中查看 CloudFront 报告。

针对 Amazon CloudFront 的合规性验证

作为多项AWS合规性计划的一部分，第三方审计员将评估 Amazon CloudFront 的安全性和合规性。其中包括 SOC、PCI、HIPAA 等。

有关特定合规性计划范围内的 AWS 服务的列表，请参阅按合规性计划提供的范围内 AWS 服务。有关一般信息，请参阅 AWS合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参见下载 AWS Artifact 中的报告。

您在使用 CloudFront 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。AWS 提供以下资源以帮助满足合规性要求：

- 安全性和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 AWS 上部署基于安全性和合规性的基准环境的步骤。
- 在 AWS 上设计符合 HIPAA 安全性与合规性要求的架构 –本白皮书介绍公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。

  AWS HIPAA 合规性计划包括 CloudFront（不包括通过 CloudFront 嵌入式 POP 交付的内容）作为符合 HIPAA 的服务。如果您已与 AWS 签订商业伙伴增订合约（BAA），则可以使用 CloudFront（不包括通过 CloudFront 嵌入式 POP 交付的内容）传送包含受保护健康信息（PHI）的内容。有关更多信息，请参阅 HIPAA 合规性。
- AWS 合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- AWS Config – 此 AWS 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub – 此 AWS 服务使用安全控件来评估资源配置和安全标准，以帮助您遵守各种合规框架。有关使用 Security Hub 评估 CloudFront 资源的更多信息，请参阅《AWS Security Hub 用户指南》中的 Amazon CloudFront 控件。

CloudFront 法规遵从性最佳实践

此部分提供了在使用 Amazon CloudFront 提供内容时的合规性最佳实践和建议。

如果您基于AWS责任共担模型运行符合 PCI 标准或符合 HIPAA 标准的工作负载，建议您记录过去 365 天的 CloudFront 使用率数据以便将来审计。要记录使用情况数据，您可以执行以下操作：

- 启用 CloudFront 访问日志。有关更多信息，请参阅 配置和使用标准日志（访问日志）。
- 捕获发送到 CloudFront API 的请求。有关更多信息，请参阅 使用 AWS CloudTrail 记录 Amazon CloudFront API 调用。

此外，请参阅以下内容以获得有关 CloudFront 如何遵循 PCI DSS 和 SOC 标准的详细信息。

支付卡行业数据安全标准 (PCI DSS)

CloudFront（不包括通过 CloudFront 嵌入式 POP 交付的内容）支持由商家或服务提供商处理、存储和传输信用卡数据，而且已经验证，符合支付卡行业（PCI）数据安全标准（DSS）。有关 PCI DSS 的更多信息，包括如何请求 AWS PCI Compliance Package 的副本，请参阅 PCI DSS 第 1 级。

作为安全最佳实践，建议您不要在 CloudFront 边缘缓存中缓存信用卡信息。例如，您可以配置源以在响应中包含 `Cache-Control:no-cache="field-name"` 标头，这些响应包含信用卡信息，例如信用卡号的后四位数以及信用卡所有者的联系信息。

系统和组织控制（SOC）

CloudFront（不包括通过 CloudFront 嵌入式 POP 交付的内容）遵循系统和组织控制（SOC）措施，包括 SOC 1、SOC 2 和 SOC 3。SOC 报告是独立的第三方检查报告，阐明AWS如何达成关键合规性控制和目标。这些审计可确保采取适当的安全措施和程序，防止出现风险，影响客户和公司数据的安全性、机密性和可用性。这些第三方审核的结果会发布在 AWS SOC 合规性网站上，您可在此查看已发布的报告，详细了解AWS为运营和合规性提供支持的控制措施。

Amazon CloudFront 中的恢复能力

AWS全球基础设施围绕 AWS 区域和可用区构建。AWS区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 AWS全球基础设施。

CloudFront 源故障转移

除了支持 AWS 全球基础设施外，Amazon CloudFront 还提供源故障转移功能，以帮助满足您的数据恢复能力需求。CloudFront 是一项全球性服务，通过称为边缘站点或节点 (POP) 的全球数据中心网络提供您的内容。如果您的内容没有在边缘站点中缓存，CloudFront 会从您已标识为源的源位置检索它，以获取内容的权威版本。

您还可以通过对 CloudFront 设置源故障转移，针对特定场景改进弹性并提升可用性。要开始操作，您需要创建一个源组，您可以在其中指定 CloudFront 的主源以及第二个源。当主源返回特定 HTTP 状态代码故障响应时，CloudFront 会自动切换到第二个源。有关更多信息，请参阅 通过 CloudFront 源失效转移来优化高可用性。

Amazon CloudFront 中的基础设施安全性

作为一项托管式服务，Amazon CloudFront 受 AWS 全球网络安全保护。有关 AWS 安全服务以及 AWS 如何保护基础设施的信息，请参阅 AWS 云安全。要按照基础设施安全最佳实践设计您的 AWS 环境，请参阅《安全性支柱 AWS Well-Architected Framework》中的 基础设施保护。

您可以使用AWS发布的 API 调用通过网络访问 CloudFront。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。

CloudFront Functions 在AWS账户之间使用高度安全的隔离屏障，确保客户环境可以安全抵御 Spectre 和 Meltdown 等侧渠道攻击。Functions 无法访问或修改属于其他客户的数据。函数在没有超线程的情况下，在专用 CPU 上以专用的单线程进程运行。在任何给定的 CloudFront 边缘站点节点 (POP) 中，CloudFront Functions 一次只为一个客户提供服务，所有客户特定的数据都会在函数执行之间清除。

故障排除

排除您在设置用于分发内容的 Amazon CloudFront 时或使用 Lambda@Edge 时可能会遇到的常见问题，并找到可能的解决方法。

主题

排查分配问题

使用此处的信息，诊断和修复您使用 Amazon CloudFront 分配设置网站或应用程序时可能遇到的证书错误、拒绝访问或其他常见问题。

主题

CloudFront 会返回 Access Denied 错误

如果您使用 Amazon S3 存储桶作为 CloudFront 分配源，您可能会在以下示例中看到“访问被拒绝”（403）错误消息。

目录

您指定了 Amazon S3 源中缺少的对象

验证您的存储桶中是否存在请求的对象。对象名称区分大小写。输入无效对象名称可能会返回“访问被拒绝”错误代码。

例如，如果您按照 CloudFront 教程创建基本分配，则可以创建一个 Amazon S3 存储桶作为源并上传 `index.html` 文件示例。

在 Web 浏览器中，如果输入 `https://d111111abcdef8.cloudfront.net/INDEX.HTML`，而不是输入 `https://d111111abcdef8.cloudfront.net/index.html`，您可能会看到类似的消息，因为 URL 路径中的 `index.html` 文件区分大小写。

```
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<RequestId>22Q367AHT7Y1ABCD</RequestId>
<HostId>
ABCDE/Vg+7PSNa/d/IfFQ8Fb92TGQ0KH0ZwG5iEKbc6+e06DdMS1ZW+ryB9GFRIVtS66rSSy6So=
</HostId>
</Error>
```

您的 Amazon S3 源缺少 IAM 权限

请确认您选择了正确的 Amazon S3 存储桶作为源域和名称。源（Amazon S3）必须拥有正确的权限。

如果您未指定正确的权限，则可能会向查看器显示以下访问被拒绝消息。

```
<Code>AccessDenied</Code>
<Message>User: arn:aws:sts::856369053181:assumed-role/OriginAccessControlRole/
EdgeCredentialsProxy+EdgeHostAuthenticationClient is not authorized to perform:
 kms:Decrypt on the resource associated with this ciphertext because the resource does
 not exist in this Region, no resource-based policies allow access, or a resource-based
 policy explicitly denies access</Message>
<RequestId>22Q367AHT7Y1ABCD/RequestId>
<HostId>
ABCDE/Vg+7PSNa/d/IfFQ8Fb92TGQ0KH0ZwG5iEKbc6+e06DdMS1ZW+ryB9GFRIVtS66rSSy6So=
</HostId>
</Error>
```

> **Note**
>
> 在此错误消息中，账户 ID 856369053181 是一个 AWS 托管账户。

当您分发来自 Amazon S3 的内容并且还使用 AWS Key Management Service（AWS KMS）服务端加密（SSE-KMS）功能时，您需要为 KMS 密钥和 Amazon S3 存储桶指定额外的 IAM 权限。您的 CloudFront 分配需要这些权限才能使用 KMS 密钥，该密钥用于加密原始 Amazon S3 存储桶。

Amazon S3 存储桶策略配置允许 CloudFront 分配检索加密对象以进行内容传输。

验证您的 Amazon S3 存储桶和 KMS 密钥权限

1. 确认您使用的 KMS 密钥与您的 Amazon S3 存储桶用于默认加密的密钥相同。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的借助 AWS KMS（SSE-KMS）指定服务器端加密。
2. 验证存储桶中的对象是否使用相同的 KMS 密钥进行了加密。您可以从 Amazon S3 存储桶中选择任何对象并检查服务器端加密设置以验证 KMS 密钥 ARN。
3. 编辑 Amazon S3 存储桶策略，授予 CloudFront 从 Amazon S3 存储桶调用 `GetObject` API 操作的权限。有关使用源访问控制的 Amazon S3 存储桶策略示例，请参阅向源访问控制授予访问 S3 存储桶的权限。
4. 编辑 KMS 密钥策略，授予 CloudFront 对 `Encrypt`、`Decrypt` 和 `GenerateDataKey*` 执行操作的权限。要与“最低权限”权限保持一致，请指定一个 `Condition` 元素，以便只有指定的 CloudFront 分配才能执行列出的操作。您可以针对现有策略自定义 AWS KMS 策略。有关 KMS 密钥策略示例，请参阅 SSE-KMS。

如果您使用的是源访问身份（OAI）而不是 OAC，则对 Amazon S3 存储桶的权限会略有不同，因为您向身份，而不是 AWS 服务 授予权限。有关更多信息，请参阅 授予源访问身份读取 Amazon S3 存储桶中文件的权限。

如果您仍然无法查看分配中的文件，请参阅我无法查看我的分配中的文件。

您使用的凭证无效或您的权限不足

如果您使用的 AWS SCT 凭证（访问密钥和私有密钥）不正确或已过期，或者您的 IAM 角色或用户缺少对 CloudFront 资源执行操作所需的权限，则可能会出现“访问被拒绝”错误消息。有关“访问被拒绝”错误消息的更多信息，请参阅《IAM 用户指南》中的排查访问被拒绝错误消息。

有关如何将 IAM 与 CloudFront 结合使用的更多信息，请参阅 适用于 Amazon CloudFront 的 Identity and Access Management。

当我尝试添加备用域名时，CloudFront 返回 InvalidViewerCertificate 错误

如果在您尝试向分配中添加备用域名（CNAME）时，CloudFront 返回 `InvalidViewerCertificate` 错误，请查看以下信息来帮助排查问题。此错误可能指示必须先解决以下问题之一，然后才能成功添加备用域名。

以下错误按照 CloudFront 检查添加备用域名的授权的顺序列出。这可以帮助您诊断问题，因为基于 CloudFront 返回的错误，您可以辨别哪些验证检查已成功完成。

没有证书附加到您的分配。

要添加备用域名 (CNAME)，您必须在分配中附加可信且有效的证书。请检查要求，获得符合要求的有效证书，将其附加到您的分配中，然后重试。有关更多信息，请参阅 使用备用域名的要求。

在您附加的证书的证书链中有过多的证书。

一个证书链中最多只能有五个证书。减少链中的证书数，然后重试。

证书链包含的一个或多个证书对于当前日期无效。

已添加的证书的证书链中有一个或多个证书无效，可能是证书迄今尚未生效，或者证书已过期。检查证书链中证书的生效日期和到期日期字段，确保基于您所列日期所有证书都有效。

附加的证书未获得信任的证书颁发机构 (CA) 的签字。

您附加到 CloudFront 以证明备用域名的证书不能是自签名证书。必须获得信任的 CA 的签名。有关更多信息，请参阅 使用备用域名的要求。

您附加的证书的格式不正确

证书中包括的域名和 IP 地址的格式以及证书本身的格式都必须遵循证书标准。

存在 CloudFront 内部错误。

CloudFront 受到内部问题阻止，无法对证书进行验证。在这种情况下，CloudFront 将返回 HTTP 500 状态代码，并且指出附加证书时 CloudFront 发生内部问题。等待几分钟，然后重试以在证书中添加备用域名。

附加的证书不涵盖您尝试添加的备用域名。

对于每个您要添加的备用域名，CloudFront 要求您附加来自信任的证书颁发机构 (CA) 且覆盖该域名的有效的 SSL/TLS 证书，以验证您有权使用它。请更新您的证书，以包含涵盖您尝试添加的 CNAME 的域名。有关在域名中使用通配符的更多信息和示例，请参阅使用备用域名的要求。

我无法查看我的分配中的文件

如果您无法查看 CloudFront 分配中的文件，请参阅下列主题了解一些常见的解决方案。

您是否同时注册了 CloudFront 和 Amazon S3？

要将 Amazon CloudFront 与 Amazon S3 源配合使用，您必须单独注册 CloudFront 和Amazon S3。有关注册 CloudFront 和 Amazon S3 的更多信息，请参阅设置。

您的 Amazon S3 存储桶和对象权限的设置是否正确？

如果您将 CloudFront 和 Amazon S3 源结合使用，对象的原始版本存储在 S3 存储桶中。将 CloudFront 与 Amazon S3 结合使用的最简单方法是让您的所有对象在 Amazon S3 中都可以公开读取。为此，您必须为上传到 Amazon S3 中的每个对象明确启用公共读取权限。

如果您的内容不是公开可读，则必须创建 CloudFront 源访问控制（OAC），以供 CloudFront 访问。有关 CloudFront 源访问控制的更多信息，请参阅the section called “限制对 Amazon Simple Storage Service 源的访问”。

对象属性和存储桶属性是独立的。您必须明确授予对 Amazon S3 存储桶中每个对象的特权。对象不会从存储桶继承属性，对象属性必须独立于存储桶进行设置。

您的备用域名 (CNAME) 配置正确吗？

如果您的域名已经具有现有的 CNAME 记录，请更新此记录或用指向分配的域名的新记录替换它。

此外，确保您的 CNAME 记录指向您分配的域名，而不是 Amazon S3 存储桶。您可确认您的 DNS 系统中的 CNAME 记录指向您分配的域名。要做到这一点，请使用 DNS 工具，如 dig。

以下示例显示对名为 `images.example.com` 的域名的 dig 请求和响应的相关部分。在 `ANSWER SECTION` 下，请参阅包含 `CNAME` 的行。如果别名记录 (CNAME) 右侧上的值是您的 CloudFront 分配域名，则您域名的 CNAME 记录设置正确。如果是您的 Amazon S3 源服务器或一些其他域名，则别名记录 (CNAME) 设置不正确。

```
[prompt]> dig images.example.com

; <<> DiG 9.3.3rc2 <<> images.example.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15917
;; flags: qr rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 2, ADDITIONAL: 0
```

```
;; QUESTION SECTION:
;images.example.com.   IN  A
;; ANSWER SECTION:
images.example.com. 10800 IN CNAME d111111abcdef8.cloudfront.net.
...
...
```

有关别名记录 (CNAME) 的更多信息，请参阅 通过添加备用域名（CNAME）使用自定义 URL。

您为 CloudFront 分配引用了正确的 URL 吗？

请确保您引用的 URL 使用 CloudFront 分配的域名（或 CNAME），而不是 Amazon S3 存储桶或自定义源。

您需要帮助来解决自定义源的问题吗？

如果您需要 AWS 帮助您解决自定义源的问题，我们可能需要检查您请求中的 `X-Amz-Cf-Id` 标头条目。如果您没有记录这些条目，您将来可能需要它。有关更多信息，请参阅 the section called “使用 Amazon EC2（或其他自定义源）”。如需进一步帮助，请访问 AWS 支持中心。

错误消息：CloudFront 正在使用证书：<certificate-id>

问题：当您尝试从 IAM 证书存储区中删除 SSL/TLS 证书时收到消息“CloudFront 正在使用证书: <certificate-id>”。

解决方案：每个 CloudFront 分配必须与默认 CloudFront 证书或自定义 SSL/TLS 证书相关联。在删除 SSL/TLS 证书前，必须轮换证书（将当前的自定义 SSL/TLS 证书替换为其他自定义 SSL/TLS 证书），或从使用自定义 SSL/TLS 证书恢复为使用默认 CloudFront 证书。要修复这一问题，请完成以下过程之一中的步骤：

- 轮换 SSL/TLS 证书
- 从自定义 SSL/TLS 证书恢复为默认 CloudFront 证书

对来自源的错误响应进行故障排除

如果 CloudFront 请求源中的对象，并且源返回 HTTP 4xx 或 5xx 状态代码，则 CloudFront 与源之间的通信存在问题。以下主题描述了某些 HTTP 状态代码的常见原因以及一些可能的解决方案。

主题

HTTP 400 状态代码（错误请求）

您的 CloudFront 分配可能会发送带有 HTTP 状态代码 400 错误请求的错误响应，以及类似于以下内容的消息：

授权标头格式错误；区域 '*<AWS Region>*' 错误；需要 '*<AWS Region>*'

例如：

授权标头格式错误；区域“us-east-1”错误；需要“us-west-2”

以下情况下可能会出现此问题：

1. 您的 CloudFront 分配的来源是一个 Amazon S3 存储桶。
2. 您将 S3 存储桶从一个 AWS 区域移动到了另一个区域。也就是说，您删除了 S3 存储桶，之后您创建了一个同名的新存储桶，但位于与原始 S3 存储桶所在位置不同的 AWS 区域。

要修复此错误，请更新您的 CloudFront 分配，以便在存储桶的当前 AWS 区域中找到 S3 存储桶。

更新 CloudFront 分配

1. 登录 AWS Management Console，并通过以下网址打开 CloudFront 控制台：https://console.aws.amazon.com/cloudfront/v4/home。
2. 选择产生此错误的分配。
3. 选择源和源组。
4. 查找您移动的 S3 存储桶的源。选中此源旁边的复选框，然后选择编辑。
5. 选择是，编辑。在选择是，编辑之前，您无需更改任何设置。

完成这些步骤后，CloudFront 将重新部署您的分配。部署分配时，您会在上次修改时间列下看到正在部署状态。部署完成后一段时间，您应停止接收 `AuthorizationHeaderMalformed` 错误响应。

HTTP 502 状态代码（无效网关）

HTTP 502 状态代码（无效网关）指示 CloudFront 因无法连接至源服务器而无法提供请求的对象。

如果您使用的是 Lambda@Edge，则问题可能是 Lambda 验证错误。如果您收到了 HTTP 502 错误以及 `NonS3OriginDnsError` 错误代码，则可能是 DNS 配置问题，导致 CloudFront 无法连接到源。

主题

CloudFront 与自定义源服务器之间的 SSL/TLS 协商失败

如果您使用自定义源并将 CloudFront 配置为要求 CloudFront 与源之间使用 HTTPS，则问题可能在于域名不匹配。在源上安装的 SSL/TLS 证书的公用名字段中包含一个域名，使用者备用名称字段中可能包含更多域名。（CloudFront 支持证书域名中的通配符。）证书中必须有一个域名与下面的一个或两个值匹配：

- 您为分配中适用源的源域指定的值。
- 如果您将 CloudFront 配置为将 `Host` 标头转发到您的源，则为该 `Host` 标头的值。有关将 `Host` 标头转发到源的更多信息，请参阅根据请求标头缓存内容。

如果域名不匹配，SSL/TLS 握手将失败，CloudFront 将返回 HTTP 状态代码 502（无效网关）并将 `X-Cache` 标头设置为 `Error from cloudfront`。

要确定证书中的域名是否与分配或 `Host` 标头中的源域匹配，可以使用在线 SSL 检查程序或 OpenSSL。如果域名不匹配，则有两个选项：

- 您为分配中适用源的源域名指定的值。
- 如果您将 CloudFront 配置为将 `Host` 标头转发到您的源，则为该 `Host` 标头的值。有关将 `Host` 标头转发到源的更多信息，请参阅根据请求标头缓存内容。

如果域名不匹配，SSL/TLS 握手将失败，CloudFront 将返回 HTTP 状态代码 502（无效网关）并将 `X-Cache` 标头设置为 `Error from cloudfront`。

要确定证书中的域名是否与分配或 `Host` 标头中的 Origin Domain Name 匹配，可以使用在线 SSL 检查程序或 OpenSSL。如果域名不匹配，则有两个选项：

- 获取包含相应域名的新 SSL/TLS 证书。

  如果您使用 AWS Certificate Manager（ACM），请参阅《AWS Certificate Manager 用户指南》中的请求证书来请求新证书。
- 更改分配配置，以使 CloudFront 不再尝试使用 SSL 来连接源。

在线 SSL 检查程序

要查找 SSL 测试工具，请在 Internet 上搜索“在线 ssl 检查程序。” 通常情况下，您指定域名，该工具将返回有关 SSL/TLS 证书的各种信息。确认在证书的公用名或使用者备用名称字段中包含您的域名。

OpenSSL

要帮助纠正来自 CloudFront 的 HTTP 502 错误，您可以使用 OpenSSL 尝试与源服务器建立 SSL/TLS 连接。如果 OpenSSL 无法建立连接，则可能表明源服务器的 SSL/TLS 配置出错。如果 OpenSSL 能够建立连接，它将返回有关源服务器证书的信息，包括证书的公用名称（`Subject CN` 字段）和使用者备用名称（`Subject Alternative Name` 字段）。

使用以下 OpenSSL 命令测试与源服务器的连接（将*##*替换为源服务器的域名，如 example.com）：

```
openssl s_client -connect origin domain name:443
```

如果满足以下条件：

- 您的源服务器支持具有多个 SSL/TLS 证书的多个域名
- 您的分配已配置为将 `Host` 标头转发到源

然后，将 `-servername` 选项添加到 OpenSSL 命令中，如以下示例所示（将 *CNAME* 替换为分配中配置的 CNAME）：

```
openssl s_client -connect origin domain name:443 -servername CNAME
```

源没有使用支持的密码/协议进行响应

CloudFront 使用密码和协议连接到源服务器。有关 CloudFront 支持的密码和协议列表，请参阅 the section called “CloudFront 与源之间受支持的协议和密码”。如果您的源在 SSL/TLS 交换中不用这些密码或协议之一进行响应，则 CloudFront 无法连接。可以使用 SSL Labs 之类的在线工具验证源是否支持密码和协议。在 Hostname 字段中键入源的域名，然后选择 Submit。查看测试中的 Common names 和 Alternative names 字段，以确定它们是否与源的域名匹配。测试完成后，在测试结果中查找 Protocols 和 Cipher Suites 部分，以确定源支持的密码和协议。将它们与 the section called “CloudFront 与源之间受支持的协议和密码” 的列表比较。

源中的 SSL/TLS 证书已过期、无效、为自签名或证书链的顺序错误

如果源服务器返回以下内容，CloudFront 将中断 TCP 连接、返回 HTTP 状态代码 502（无效网关），并将 `X-Cache` 标头设置为 `Error from cloudfront`：

- 过期的证书
- 无效的证书
- 自签名证书
- 证书链的顺序错误

> **Note**
>
> 如果不存在完整的证书链（包括中间证书），CloudFront 将中断 TCP 连接。

有关在自定义源服务器上安装 SSL/TLS 证书的信息，请参阅 the section called “要求通过 HTTPS 连接到自定义源”。

源在“源设置”中的指定端口上没有响应

在 CloudFront 分配中创建源时，您可以为 HTTP 和 HTTPS 流量设置 CloudFront 用于连接到源的端口。默认情况下，这些端口为 TCP 80/443。您可以选择修改这些端口。如果源出于任何原因拒绝这些端口上的流量，或者后端服务器在这些端口上无响应，CloudFront 将无法连接。

要解决这些问题，请检查您的基础设施中运行的任何防火墙并确认它们未阻止支持的 IP 范围。有关更多信息，请参阅《Amazon Web Services 一般参考》中的 AWS IP 地址范围。此外，还要验证您的 Web 服务器是否正在源中运行。

Lambda 验证错误

如果使用的是 Lambda@Edge，则 HTTP 502 状态代码可能指示未正确设置 Lambda 函数响应格式或包含无效的内容。有关排查 Lambda@Edge 错误的更多信息，请参阅测试和调试 Lambda@Edge 函数。

DNS 错误（**NonS3OriginDnsError**）

错误代码为 NonS3OriginDnsError 的 HTTP 502 错误表示存在 DNS 配置问题，导致 CloudFront 无法连接到源。如果您从 CloudFront 收到此错误，请确保源的 DNS 配置正确且有效。

当 CloudFront 收到针对已过期或不在缓存中的对象的请求时，它会向源发出请求，以获取该对象。为了成功向源发出请求，CloudFront 会对源域执行 DNS 解析。如果您的域的 DNS 服务出现问题时，CloudFront 将无法解析域名以获取 IP 地址，从而导致 HTTP 502 错误 (NonS3OriginDnsError)。要修复此问题，请联系您的 DNS 提供商，或者，如果您使用的是 Amazon Route 53，请参阅为什么我无法访问使用 Route 53 DNS 服务的网站？

要进一步解决该问题，请确保源的根域或顶级域（如 example.com）的权威名称服务器运行正常。您可以通过 dig 或 nslookup等工具，使用以下命令查找顶级源的名称服务器：

```
dig OriginAPEXDomainName NS +short
```

```
nslookup -query=NS OriginAPEXDomainName
```

如果您有名称服务器的名称，请使用以下命令针对这些名称查询源的域名，以确保每个查询都会得到响应：

```
dig OriginDomainName @NameServer
```

```
nslookup OriginDomainName NameServer
```

> **Important**
>
> 确保使用连接到公共互联网的计算机执行此 DNS 故障排除。CloudFront 使用互联网上的公有 DNS 解析源域，因此在类似环境中进行故障排除非常重要。

如果您的源是一个子域，其 DNS 权限被委托给与根域不同的域名服务器，请确保为该子域正确配置域名服务器 (NS) 和授权起始点 (SOA) 记录。您可以使用与上述示例类似的命令来检查这些记录。

有关 DNS 的更多信息，请参阅 Amazon Route 53 文档中的域名系统 (DNS) 概念。

HTTP 503 状态代码（服务不可用）

HTTP 503 状态代码（服务不可用）通常表示源服务器存在性能问题。在极少数情况下，该代码表示由于边缘站点中的资源限制，CloudFront 暂时无法满足请求。

如果您使用的是 Lambda@Edge 或 CloudFront Functions，则问题可能是执行错误或超出 Lambda@Edge 限制错误。

主题

源服务器没有足够容量来支持请求速率

当原始服务器不可用或无法处理传入请求时，将返回 HTTP 503 状态代码（服务不可用）。然后，CloudFront 会将错误返回给用户。要解决该问题，请尝试以下解决方案：

- 如果您使用 Amazon S3 作为原始服务器：
  - 对于每个分区的 Amazon S3 前缀，您可以每秒执行至少 3500 个 PUT/COPY/POST/DELETE 请求或 5500 个 GET/HEAD 请求。当 Amazon S3 返回 503 减速响应时，这通常表示针对特定 Amazon S3 前缀的请求速率过高。

    由于请求速率针对的是 S3 存储桶中的各个前缀，因此应将对象分布在多个前缀中。随着前缀上的请求速率逐渐提高，Amazon S3 会纵向扩展以分别处理每个前缀的请求。因此，存储桶所处理的总请求速率是前缀数量的倍数。
  - 有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的性能 Amazon S3 优化。
- 如果您为原始服务器使用弹性负载均衡，则：
  - 确保您的后端实例可以响应运行状况检查。
  - 确保您的负载均衡器和后端实例可以处理负载。

  有关更多信息，请参阅：
  - 在使用经典负载均衡器时，如何对返回的 503 错误进行故障排除？

  - 如何对应用程序负载均衡器返回的 503（服务不可用）错误进行故障排除？
- 使用自定义源时：
  - 请检查应用程序日志，以确保您的源具有足够资源，如内存、CPU 和磁盘大小。
  - 如果使用 Amazon EC2 作为后端，请确保实例类型具有适当的资源来满足传入请求。有关更多信息，请参阅《Amazon EC2 用户指南》中的实例类型。
- 使用 API Gateway 时：
  - 在 API Gateway API 无法接收响应时，此错误与后端集成有关。后端服务器可能：
    - 负载已超出容量，无法处理新的客户端请求。
    - 正在进行临时维护。
  - 要解决此错误，请查看您的 API Gateway 应用程序日志，以确定后端容量、集成或其他方面是否存在问题。

由于边缘站点中的资源限制，CloudFront 导致发生错误

在极少数情况下，CloudFront 无法将请求路由到下一个可用的最佳边缘站点并因此无法满足请求，此时您将收到此错误。当您对 CloudFront 分配执行负载测试时，此错误很常见。为帮助防止发生此情况，请遵循the section called “对 CloudFront 进行负载测试” 指南，以避免 503（超出容量）错误。

如果在生产环境中发生这种情况，请联系 AWS Support。

Lambda@Edge 或 CloudFront Function 执行错误

如果您使用的是 Lambda@Edge 或 CloudFront Functions，则 HTTP 503 状态代码可能指示您的函数返回了执行错误。

有关如何识别和解决 Lambda@Edge 错误的详细信息，请参阅测试和调试 Lambda@Edge 函数。

有关对 CloudFront Functions 进行测试的更多信息，请参阅测试函数。

Lambda@Edge 超出限制

如果您使用的是 Lambda@Edge，则 HTTP 503 状态代码可能表明 Lambda 返回了错误。此错误可能是由于下列原因之一导致的：

- 函数执行数量超过了 Lambda 为限制 AWS 区域中的执行数量而设置的配额之一（并发执行或调用频率）。
- 函数超出了 Lambda 函数超时配额。

有关 Lambda@Edge 配额的更多信息，请参阅 有关 Lambda@Edge 的配额。有关如何识别和解决 Lambda@Edge 错误的详细信息，请参阅the section called “测试和调试”。您还可以查看《AWS Lambda 开发人员指南》中的 Lambda 服务配额。

HTTP 504 状态代码（网关超时）

HTTP 504 状态代码（网关超时）指示当 CloudFront 将请求转发至源后（因为请求的对象不在边缘缓存中），发生了以下情况之一：

- 源将 HTTP 504 状态代码返回到 CloudFront。
- 源在请求过期前未响应。

如果流量被防火墙或安全组阻止进入源，或者无法在 Internet 上访问源，CloudFront 将返回 HTTP 504 状态代码。请首先检查是否存在这些问题。然后，如果访问没有问题，请探究应用程序延迟和服务器超时，以帮助您确定并解决问题。

主题

- 将您的源服务器上的防火墙设置为允许 CloudFront 流量
- 将您的源服务器上的安全组设置为允许 CloudFront 流量
- 使自定义源服务器在 Internet 上可访问
- 查找并修复源服务器上来自应用程序的延迟响应

将您的源服务器上的防火墙设置为允许 CloudFront 流量

如果您的源服务器上的防火墙阻止 CloudFront 流量，CloudFront 将返回 HTTP 504 状态代码，所以在检查其他问题之前，务必确保不是这个问题的原因。

您用于确定这是否是与您的防火墙相关的问题的方法取决于源服务器使用的系统：

- 如果您使用的是 Linux 服务器上的 IPTable 防火墙，则可以搜索帮助您使用 IPTable 的工具和信息。
- 如果您使用的是 Windows 服务器上的 Windows 防火墙，请参阅 Microsoft 文档中的添加或编辑防火墙规则。

当您评估源服务器上的防火墙配置时，基于发布的 IP 地址范围查找阻止来自 CloudFront 边缘站点的流量的任何防火墙或安全规则。有关更多信息，请参阅 CloudFront 边缘服务器的位置和 IP 地址范围。

如果 CloudFront IP 地址范围已允许连接至您的源服务器，请确保更新您的服务器的安全规则以合并更改。您可以订阅 Amazon SNS 主题并且在更新 IP 地址范围文件时接收通知。收到该通知后，您可以使用代码检索该文件、解析文件并根据您的当地环境做出相应调整。有关更多信息，请参阅 AWS 新闻博客上的通过 Amazon SNS 订阅 AWS 公共 IP 地址范围。

将您的源服务器上的安全组设置为允许 CloudFront 流量

如果您的源使用 Elastic Load Balancing，请检查 ELB 安全组并确保安全组允许来自 CloudFront 的入站流量。

您还可以使用 AWS Lambda 自动更新您的安全组，以允许来自 CloudFront 的入站流量。

使自定义源服务器在 Internet 上可访问

如果 CloudFront 由于自定义源服务器未在 Internet 上公开而无法对其进行访问，CloudFront 将返回 HTTP 504 错误。

CloudFront 边缘站点通过 Internet 连接到源服务器。如果自定义源位于私有网络上，CloudFront 则无法访问它。因此，您无法使用私有服务器（包括内部经典负载均衡器）作为 CloudFront 的源服务器。

要检查互联网流量是否可以连接到原始服务器，请运行以下命令（其中 *OriginDomainName* 是您的服务器的域名）：

对于 HTTPS 流量：

- nc -zv *OriginDomainName* 443
- telnet *OriginDomainName* 443

对于 HTTP 流量：

- nc -zv *OriginDomainName* 80
- telnet *OriginDomainName* 80

查找并修复源服务器上来自应用程序的延迟响应

服务器超时通常是应用程序响应时间过长或超时值设置得过低的结果。

帮助避免 HTTP 504 错误的快速解决方法是为您的分配设置一个较高的 CloudFront 超时值。但是，建议您首先确保解决与应用程序和源服务器相关的任何性能和延迟问题。然后，您可以设置一个合理的超时值，以帮助防止 HTTP 504 错误并向用户提供良好的响应能力。

以下是您要查找性能问题并进行更正可以采取的步骤的概述：

1. 测量您的 Web 应用程序的典型和高负载延迟 (响应能力)。
2. 添加其他资源，如 CPU 或内存 (如果需要)。采取解决问题的其他步骤，如优化数据库查询以适应高负载场景。
3. 如果需要，调整您的 CloudFront 分配的超时值。

以下是关于每个步骤的详细信息。

测量典型和高负载延迟

要确定一个或多个后端 Web 应用程序服务器是否遇到高延迟，请在每个服务器上运行以下 Linux curl 命令：

```
curl -w "Connect time: %{time_connect} Time to first byte: %{time_starttransfer} Total
 time: %{time_total} \n" -o /dev/null https://www.example.com/yourobject
```

> **Note**
>
> 如果您在服务器上运行的是 Windows，则可以搜索并下载用于 Windows 运行类似命令的 curl。

您在测量并评估在您的服务器上运行的应用程序的延迟时，请注意以下事项：

- 延迟值是相对于每个应用程序的。但是，合理的做法是第一个字节的时间采用毫秒而不是秒或更大的单位。
- 如果您在正常负载下测量应用程序延迟且结果正常，请注意查看器在高负载下可能仍会遇到超时。当存在高需求时，服务器可能已延迟响应或根本未响应。为了帮助防止高负载延迟问题，请检查您的服务器的资源，如 CPU、内存和磁盘读写情况，以确保您的服务器具有针对高负载进行扩展的容量。

  您可以运行以下 Linux 命令来检查 Apache 进程所使用的内存：

  ```
  watch -n 1 "echo -n 'Apache Processes: ' && ps -C apache2 --no-headers |
  wc -l && free -m"
  ```
- 服务器上的 CPU 使用率高可能会大幅降低应用程序的性能。如果您对后端服务器使用的是 Amazon EC2 实例，请检查服务器的 CloudWatch 指标以查看 CPU 利用率。有关更多信息，请参阅 Amazon

CloudWatch 用户指南。或者，如果您使用的是自己的服务器，请参阅该服务器的有关如何检查 CPU 使用率的说明的帮助文档。

- 检查在高负载下是否存在其他潜在问题，如存在大量请求时数据库查询运行缓慢。

添加资源并优化服务器和数据库

在您评估应用程序和服务器的响应能力后，请确保您有用于典型流量和高负载情况的足够资源：

- 如果您有自己的服务器，请确保它具有足够的 CPU、内存和磁盘空间来处理查看器请求 (根据您的评估)。
- 如果使用 Amazon EC2 实例作为后端服务器，请确保实例类型具有适当的资源来满足传入请求。有关更多信息，请参阅《Amazon EC2 用户指南》中的实例类型。

此外，请考虑以下帮助避免超时的优化步骤：

- 如果 curl 命令返回的第一个字节的时间值似乎很高，请采取改进您的应用程序性能的步骤。提高应用程序响应能力反过来将有助于减少超时错误。
- 优化数据库查询以确保它们可以处理大量请求，而不会降低性能。
- 在您的后端服务器上设置保持连接 (持久性) 连接。该选项有助于避免在必须为后续请求或用户重新建立连接时发生的延迟。
- 如果使用 ELB 作为源，请通过查看以下知识中心文章中的建议了解如何减少延迟：如何排查我的 ELB 经典负载均衡器上的高延迟问题？

如果需要，请调整 CloudFront 超时值

如果您已评估并解决低应用程序性能、源服务器容量及其他问题，但查看器仍遇到 HTTP 504 错误，则您应考虑更改在您的分配中为源响应超时指定的时间。有关更多信息，请参阅 the section called “响应超时（仅自定义源）”。

对 CloudFront 进行负载测试

传统的负载测试方法不能很好地与 CloudFront 一起使用，因为 CloudFront 使用 DNS 平衡跨越地理上分散的边缘站点的负载和每个边缘站点内的负载。当客户端从 CloudFront 请求内容时，客户端收到包含一组 IP 地址的 DNS 响应。如果您采用的测试方法是只向 DNS 返回的其中一个 IP 地址发送请求，那么您测试的仅仅是一个 CloudFront 边缘站点中的一小部分资源，这并不能准确体现实际的流量规

律。根据所请求的数据量，以这种方式进行测试可能会造成这一小部分 CloudFront 服务器超载且性能下降。

CloudFront 的设计旨在扩展在多个地理区域具有不同客户端 IP 地址和不同 DNS 分解器的查看器。要执行准确评估 CloudFront 性能的负载测试，建议您执行以下所有操作：

- 从多个地理区域发送客户端请求。
- 配置您的测试，让每个客户端发出独立的 DNS 请求。然后，每个客户端从 DNS 接收一组不同的 IP 地址。
- 对于每个发出请求的客户端，将客户端请求分布在 DNS 返回的一组 IP 地址上。这样可以确保将负载分布在 CloudFront 边缘站点中的多个服务器上。

> **注意**
>
> - 不允许对包含 Lambda@Edge 查看器请求或查看器响应触发器的缓存行为进行负载测试。
> - 不允许对已启用 Origin Shield 的源进行负载测试。

配额

CloudFront 受以下配额约束。

主题

常规配额

实体	默认配额
每次分发的数据传输速率	150 Gbps 请求更高的配额
每次分发每秒的请求数	250,000 请求更高的配额
可添加到分配的标签	50

实体	默认配额
	请求更高的配额
可按分配提供的文件	没有配额
请求或源响应的最大长度，包括标头和查询字符串，但不包括正文内容	20,480 字节
URL 的最大长度	8,192 字节

分配的一般配额

实体	默认配额
每个分配的备用域名（CNAME） 有关更多信息，请参阅 通过添加备用域名（CNAME）使用自定义 URL。	100 请求更高的配额
每个分配的缓存行为	25 请求更高的配额
每个源的连接尝试次数 有关更多信息，请参阅 连接尝试次数。	1-3
每个源的连接超时 有关更多信息，请参阅 连接超时。	1-10 秒
每个 AWS 账户的分发 有关更多信息，请参阅 创建分配。	200 请求更高的配额
基于源访问控制的分配	100 请求更高的配额
文件压缩：CloudFront 压缩的文件大小的范围	1,000-10,000,000 个字节

实体	默认配额
有关更多信息，请参阅 提供压缩文件。	
每个源的保持连接超时 有关更多信息，请参阅 源保持连接超时（仅自定义源）。	1-60 秒 请求更高的配额
每个 HTTP GET 响应的最大可缓存文件大小。 只缓存对于 HTTP GET 的响应。不缓存对于 POST 或 PUT 的响应。	50 GB
每个 AWS 账户的源访问控制	100
每个 AWS 账户的源访问身份	100 请求更高的配额
每个分配的源	25 请求更高的配额
每个分配的源组	10 请求更高的配额
每个源的响应超时 有关更多信息，请参阅 响应超时（仅自定义源）。	1-60 秒 请求更高的配额
每个 AWS 账户的暂存分发 有关更多信息，请参阅 the section called “使用持续部署来安全地测试更改”。	20 请求更高的配额

策略的一般配额

实体	默认配额
每个 AWS 账户的缓存策略	20

实体	默认配额
	请求更高的配额
与相同缓存策略关联的分配	100
每个缓存策略的查询字符串	10 请求更高的配额
每个缓存策略的标头	10 请求更高的配额
每个缓存策略的 Cookie	10 请求更高的配额
缓存策略中所有查询字符串、标头和 cookie 名称的总长度	1024
每个 AWS 账户的源请求策略	20 请求更高的配额
与相同源请求策略关联的分配	100
每个源请求策略的查询字符串	10 请求更高的配额
每个源请求策略的标头	10 请求更高的配额
每个源请求策略的 Cookie	10 请求更高的配额
源请求策略中所有查询字符串、标头和 cookie 名称的总长度	1024
每个 AWS 账户的响应标头策略	20 请求更高的配额

实体	默认配额
与相同响应标头策略关联的分配	100 请求更高的配额
每个响应标头策略的自定义标头	10 请求更高的配额
每个 AWS 账户的持续部署策略	20 请求更高的配额

CloudFront Functions 的配额

实体	默认配额
每个 AWS 账户的函数	100
最大函数大小	10 KB 请求更高的配额
最大函数内存	2MB
与相同函数关联的分配	100

除了这些配额外，在使用 CloudFront Functions 时还有一些其他限制。有关更多信息，请参阅 对 CloudFront Functions 的限制。

有关键值存储的限额

实体	默认配额
键值对中键的最大大小	512 字节

实体	默认配额
键值对中值的最大大小	1KB
您可以在单个 API 请求中更新的最大键值对数量	50 个键或 3MB 有效负载，以先到达者为准
单个键值存储的最大大小	5MB
单个键值存储可以关联的最大函数数量	10
每个函数的最大键值存储数量	1
每个账户的最大键值存储数量	50 请求更高的配额

有关 Lambda@Edge 的配额

此部分中的配额适用于 Lambda@Edge。这些配额是默认 AWS Lambda 配额（同样适用）之外的配额。有关 Lambda 配额，请参阅 AWS Lambda 开发人员指南中的配额。

> **Note**
>
> Lambda 将在您的 AWS 账户配额内，根据增加的流量动态扩展容量。有关更多信息，请参阅 AWS Lambda开发人员指南中的函数扩展。

常规配额

实体	默认配额
每个可以具有 Lambda@Edge 函数的 AWS 账户的分发	500 请求更高的配额
每个分配的 Lambda@Edge 函数	100

实体	默认配额
	请求更高的配额
每秒请求数	10,000（在每个 AWS 区域） 请求更高的配额
并发执行 有关更多信息，请参阅 AWS Lambda开发人员指南中的函数扩展。	1000（在每个 AWS 区域） 请求更高的配额
与相同函数关联的分配	500

因事件类型而不同的配额

实体	查看器请求和查看器响应事件	源请求和源响应事件
函数内存大小	128MB	与 Lambda 配额相同
函数超时。函数可以对 AWS 区域中的 Amazon S3 存储桶、DynamoDB 表或 Amazon EC2 实例等资源进行网络调用。	5 秒	30 秒
Lambda 函数生成的响应的大小，包括标头和正文	40 KB	1MB
Lambda 函数和包含的所有库压缩后的最大大小	1MB	50 MB

除这些配额外，请注意在使用 Lambda@Edge 函数时还有一些其他限制。有关更多信息，请参阅 对 Lambda@Edge 的限制。

SSL 证书的配额

实体	默认配额
当使用专用 IP 地址提供 HTTPS 请求时每个 AWS 账户的 SSL 证书数（当使用 SNI 提供 HTTPS 请求时没有配额） 有关更多信息，请参阅 将 HTTPS 与 CloudFront 结合使用。	2 请求更高的配额
可与 CloudFront 分配关联的 SSL 证书	1

如果您的 SSL 证书专门用于查看器与 CloudFront 之间的 HTTPS 通信，并且如果您使用 AWS Certificate Manager（ACM）或 IAM 证书存储来预置或导入证书，则需要遵循其他的配额。有关更多信息，请参阅 在 CloudFront 中使用 SSL/TLS 证书的配额（仅在查看器和 CloudFront 之间使用 HTTPS）。

您可以导入到 AWS Certificate Manager（ACM）或上传至 AWS Identity and Access Management（IAM）的 SSL 证书的数量也存在配额。有关更多信息，请参阅 增加 SSL/TLS 证书的配额。

有关失效的配额

实体	默认配额
文件失效：活动失效请求中允许的文件的最大数量，不包括通配符失效 有关更多信息，请参阅 使文件失效以删除内容。	3000
文件失效：允许的活动通配符失效的最大数量	15
文件失效：一个通配符失效可处理的文件的最大数量	没有配额

有关密钥组的配额

实体	默认配额
单个密钥组中的公有密钥数	5 请求更高的配额
与单个缓存行为关联的密钥组数	4 请求更高的配额
每个 AWS 账户的密钥组	10 请求更高的配额
与单个密钥组关联的分配数	100 请求更高的配额

WebSocket 连接配额

实体	默认配额
源响应超时（空闲超时）	10 分钟 如果 CloudFront 未检测到过去 10 分钟内从源发送到客户端的任何字节，则假定连接处于空闲状态并关闭连接。

对字段级加密的配额

实体	默认配额
可加密字段的最大长度	16 KB

实体	默认配额
有关更多信息，请参阅 使用字段级加密帮助保护敏感数据。	
配置字段级加密时请求正文中的最大字段数	10
配置字段级加密时请求正文的最大长度	1 MB
可与一个 AWS 账户关联的最大字段级加密配置数量	10
可与一个 AWS 账户关联的最大字段级加密配置文件数量	10
可添加到一个 AWS 账户的最大公有密钥数	10
可在一个配置文件中指定的要加密的最大字段数	10
可与字段级加密配置关联的最大 CloudFront 分配数目	20
可包含在字段级加密配置中的最大查询参数配置文件映射数目	5

Cookies 的配额（旧缓存设置）

这些配额适用于 CloudFront 的旧缓存设置。我们建议使用缓存策略或源请求策略，而不是旧设置。

实体	默认配额
每个缓存行为的 Cookies 有关更多信息，请参阅 根据 Cookie 缓存内容。	10 请求更高的配额
Cookie 名称的字节总数（如果将 CloudFront 配置为将所有 Cookie 转发给源，则不适用）	512 减去 Cookies 的数量

查询字符串的配额（旧缓存设置）

这些配额适用于 CloudFront 的旧缓存设置。我们建议使用缓存策略或源请求策略，而不是旧设置。

实体	默认配额
查询字符串中的最大字符数	128 个字符
同一参数中的所有查询字符串的最大字符数总计	512 个字符
每个缓存行为的查询字符串数 有关更多信息，请参阅 根据查询字符串参数缓存内容。	10 请求更高的配额

标头的配额

实体	默认配额
每个缓存行为的标头（旧缓存设置） 有关更多信息，请参阅 the section called “根据请求标头缓存内容”。	10 请求更高的配额
自定义标头：可将 CloudFront 配置为添加到源请求的自定义标头的最大数目 有关更多信息，请参阅 the section called “向源请求添加自定义标头”。	10 请求更高的配额
自定义标头：可添加到一个响应标头策略的最大自定义标头数	10 请求更高的配额
自定义标头：标头名称的最大长度	256 个字符
自定义标头：标头值的最大长度	1,783 个字符
自定义标头：所有标头值和名称加在一起的最大长度	10,240 个字符
`Content-Security-Policy` 标头的值的最大长度	1,783 个字符 请求更高的配额

使用 AWS SDK 的 CloudFront 代码示例

以下代码示例演示如何将 CloudFront 与 AWS 软件开发工具包（SDK）结合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景 是展示如何通过在同一服务中调用多个函数来完成特定任务的代码示例。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

使用 AWS SDK 对 CloudFront 执行的操作

以下代码示例演示如何使用 AWS SDK 来执行单个 CloudFront 操作。这些代码节选将调用 CloudFront API，是必须在上下文中运行的大型程序的代码节选。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon CloudFront API 参考。

示例

- 将 CreateDistribution 与 AWS SDK 或 CLI 配合使用
- 将 CreateFunction 与 AWS SDK 或 CLI 配合使用
- 将 CreateInvalidation 与 AWS SDK 或 CLI 配合使用
- 将 CreateKeyGroup 与 AWS SDK 或 CLI 配合使用
- 将 CreatePublicKey 与 AWS SDK 或 CLI 配合使用
- 将 DeleteDistribution 与 AWS SDK 或 CLI 配合使用
- 将 GetCloudFrontOriginAccessIdentity 与 AWS SDK 或 CLI 配合使用
- 将 GetCloudFrontOriginAccessIdentityConfig 与 AWS SDK 或 CLI 配合使用
- 将 GetDistribution 与 AWS SDK 或 CLI 配合使用
- 将 GetDistributionConfig 与 AWS SDK 或 CLI 配合使用
- 将 ListCloudFrontOriginAccessIdentities 与 AWS SDK 或 CLI 配合使用
- 将 ListDistributions 与 AWS SDK 或 CLI 配合使用
- 将 UpdateDistribution 与 AWS SDK 或 CLI 配合使用

将 `CreateDistribution` 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateDistribution`。

CLI

AWS CLI

创建 CloudFront 分配

以下示例使用命令行参数为名为 `awsexamplebucket` 的 S3 存储桶创建分配，并将 `index.html` 指定为默认根对象：

```
aws cloudfront create-distribution \
    --origin-domain-name awsexamplebucket.s3.amazonaws.com \
    --default-root-object index.html
```

您可以在 JSON 文件中提供分配配置，而不必使用命令行参数，如以下示例所示：

```
aws cloudfront create-distribution \
    --distribution-config file://dist-config.json
```

文件 dist-config.json 是当前文件夹中包含以下内容的 JSON 文档：

```
{
    "CallerReference": "cli-example",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                "DomainName": "awsexamplebucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
```

```
            },
            "Headers": {
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
```

```
    "Logging": {
        "Enabled": false,
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": true,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

无论您使用命令行参数还是 JSON 文件提供分配信息，输出都相同：

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EMLARXS9EXAMPLE",
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-11-22T00:55:15.705Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
```

```
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
```

```
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
```

```
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CreateDistribution。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

以下示例使用 Amazon Simple Storage Service (Amazon S3) 桶作为内容来源。

创建分配后，该代码会创建一个 CloudFrontWaiter，等待分配部署完成后再返回该分配

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.ItemSelection;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import java.time.Instant;

public class CreateDistribution {

        private static final Logger logger =
 LoggerFactory.getLogger(CreateDistribution.class);

        public static Distribution createDistribution(CloudFrontClient
 cloudFrontClient, S3Client s3Client,
                        final String bucketName, final String keyGroupId, final
 String originAccessControlId) {

                final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                                .get("x-amz-bucket-region").get(0);
                final String originDomain = bucketName + ".s3." + region +
 ".amazonaws.com";
                String originId = originDomain; // Use the originDomain value for
 the originId.

                // The service API requires some deprecated methods, such as
                // DefaultCacheBehavior.Builder#minTTL and #forwardedValue.
                CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                                .distributionConfig(b1 -> b1
                                                .origins(b2 -> b2
                                                                .quantity(1)
                                                                .items(b3 -> b3

 .domainName(originDomain)

 .id(originId)

 .s3OriginConfig(builder4 -> builder4

               .originAccessIdentity(

                               ""))

 .originAccessControlId(

               originAccessControlId)))
                                                .defaultCacheBehavior(b2 -> b2
```

```
.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)
                                                                .minTTL(200L)

.forwardedValues(b5 -> b5

.cookies(cp -> cp

              .forward(ItemSelection.NONE))

.queryString(true))

.trustedKeyGroups(b3 -> b3

.quantity(1)

.items(keyGroupId)

.enabled(true))

.allowedMethods(b4 -> b4

.quantity(2)

.items(Method.HEAD, Method.GET)

.cachedMethods(b5 -> b5

              .quantity(2)

              .items(Method.HEAD,

                            Method.GET))))
                                          .cacheBehaviors(b -> b
                                                          .quantity(1)
                                                          .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(

              ViewerProtocolPolicy.ALLOW_ALL)
```

```
.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

              .quantity(1)

              .items(keyGroupId)

              .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

              .cookies(cp -> cp

                            .forward(ItemSelection.NONE))

              .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

              .items(Method.HEAD,

                            Method.GET)

              .cachedMethods(b6 -> b6

                            .quantity(2)

                            .items(Method.HEAD,

                                          Method.GET)))))
                                           .enabled(true)
                                           .comment("Distribution built with
java")

.callerReference(Instant.now().toString())));

               final Distribution distribution =
createDistResponse.distribution();
               logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
distribution.domainName(),
```

```
                                distribution.id());
                logger.info("Waiting for distribution to be deployed ...");
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                                        .matched();
                        responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                        distribution.id());
                }
                return distribution;
        }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 CreateDistribution。

PowerShell

适用于 PowerShell 的工具

示例 1：创建配置了日志记录和缓存的基本 CloudFront 分配。

```
$origin = New-Object Amazon.CloudFront.Model.Origin
$origin.DomainName = "ps-cmdlet-sample.s3.amazonaws.com"
$origin.Id = "UniqueOrigin1"
$origin.S3OriginConfig = New-Object Amazon.CloudFront.Model.S3OriginConfig
$origin.S3OriginConfig.OriginAccessIdentity = ""
New-CFDistribution `
      -DistributionConfig_Enabled $true `
      -DistributionConfig_Comment "Test distribution" `
      -Origins_Item $origin `
      -Origins_Quantity 1 `
      -Logging_Enabled $true `
      -Logging_IncludeCookie $true `
      -Logging_Bucket ps-cmdlet-sample-logging.s3.amazonaws.com `
      -Logging_Prefix "help/" `
```

```
    -DistributionConfig_CallerReference Client1 `
    -DistributionConfig_DefaultRootObject index.html `
    -DefaultCacheBehavior_TargetOriginId $origin.Id `
    -ForwardedValues_QueryString $true `
    -Cookies_Forward all `
    -WhitelistedNames_Quantity 0 `
    -TrustedSigners_Enabled $false `
    -TrustedSigners_Quantity 0 `
    -DefaultCacheBehavior_ViewerProtocolPolicy allow-all `
    -DefaultCacheBehavior_MinTTL 1000 `
    -DistributionConfig_PriceClass "PriceClass_All" `
    -CacheBehaviors_Quantity 0 `
    -Aliases_Quantity 0
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 CreateDistribution。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `CreateFunction` 与 AWS SDK 或 CLI 配合使用

以下代码示例演示了如何使用 `CreateFunction`。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionRequest;
import software.amazon.awssdk.services.cloudfront.model.CreateFunctionResponse;
import software.amazon.awssdk.services.cloudfront.model.FunctionConfig;
```

```
import software.amazon.awssdk.services.cloudfront.model.FunctionRuntime;
import java.io.InputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateFunction {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <functionName> <filePath>

                Where:
                    functionName - The name of the function to create.\s
                    filePath - The path to a file that contains the application
 logic for the function.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        String funArn = createNewFunction(cloudFrontClient, functionName,
 filePath);
        System.out.println("The function ARN is " + funArn);
        cloudFrontClient.close();
    }
```

```
    public static String createNewFunction(CloudFrontClient cloudFrontClient,
String functionName, String filePath) {
        try {
            InputStream fileIs =
CreateFunction.class.getClassLoader().getResourceAsStream(filePath);
            SdkBytes functionCode = SdkBytes.fromInputStream(fileIs);

            FunctionConfig config = FunctionConfig.builder()
                    .comment("Created by using the CloudFront Java API")
                    .runtime(FunctionRuntime.CLOUDFRONT_JS_1_0)
                    .build();

            CreateFunctionRequest functionRequest =
CreateFunctionRequest.builder()
                    .name(functionName)
                    .functionCode(functionCode)
                    .functionConfig(config)
                    .build();

            CreateFunctionResponse response =
cloudFrontClient.createFunction(functionRequest);
            return response.functionSummary().functionMetadata().functionARN();

        } catch (CloudFrontException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 CreateFunction。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateInvalidation** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateInvalidation。

CLI

AWS CLI

为 CloudFront 分配创建失效

以下 create-invalidation 示例为指定 CloudFront 分配中的指定文件创建失效：

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
    --paths "/example-path/example-file.jpg" "/example-path/example-file2.png"
```

输出：

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I1JLWSDAP8FU89",
    "Invalidation": {
        "Id": "I1JLWSDAP8FU89",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:24:51.407Z",
        "InvalidationBatch": {
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file2.png",
                    "/example-path/example-file.jpg"
                ]
            },
            "CallerReference": "cli-1575570291-670203"
        }
    }
}
```

在前面的示例中，AWS CLI 自动生成了一个随机 CallerReference。要指定自己的 CallerReference，或者为了避免将失效参数作为命令行参数传递，可以使用 JSON 文件。以下示例通过在名为 inv-batch.json 的 JSON 文件中提供失效参数，从而为两个文件创建失效：

```
aws cloudfront create-invalidation \
    --distribution-id EDFDVBD6EXAMPLE \
```

```
    --invalidation-batch file://inv-batch.json
```

inv-batch.json 的内容：

```
{
    "Paths": {
        "Quantity": 2,
        "Items": [
            "/example-path/example-file.jpg",
            "/example-path/example-file2.png"
        ]
    },
    "CallerReference": "cli-example"
}
```

输出：

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/
EDFDVBD6EXAMPLE/invalidation/I2J0I21PCUYOIK",
    "Invalidation": {
        "Id": "I2J0I21PCUYOIK",
        "Status": "InProgress",
        "CreateTime": "2019-12-05T18:40:49.413Z",
        "InvalidationBatch": {
            "Paths": {
                "Quantity": 2,
                "Items": [
                    "/example-path/example-file.jpg",
                    "/example-path/example-file2.png"
                ]
            },
            "CallerReference": "cli-example"
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CreateInvalidation。

PowerShell

适用于 PowerShell 的工具

示例 1：此示例在 ID 为 EXAMPLENSTXAXE 的分配上创建一个新的失效。CallerReference 是用户选择的唯一 ID；在本例中，使用代表 2019 年 5 月 15 日上午 9:00 的时间戳。$Paths 变量存储了用户不希望将其作为分配缓存一部分的图像和媒体文件的三个路径。-Paths_Quantity 参数值是在 -Paths_Item 参数中指定的路径总数。

```
$Paths = "/images/*.gif", "/images/image1.jpg", "/videos/*.mp4"
New-CFInvalidation -DistributionId "EXAMPLENSTXAXE" -
InvalidationBatch_CallerReference 20190515090000 -Paths_Item $Paths -
Paths_Quantity 3
```

输出：

```
Invalidation                          Location

------------                          --------

Amazon.CloudFront.Model.Invalidation https://cloudfront.amazonaws.com/2018-11-05/
distribution/EXAMPLENSTXAXE/invalidation/EXAMPLE8NOK9H
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell API 参考》中的 CreateInvalidation。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateKeyGroup** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示了如何使用 CreateKeyGroup。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

密钥组需要至少一个用于验证签名 URL 或 Cookie 的公有密钥。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;

import java.util.UUID;

public class CreateKeyGroup {
    private static final Logger logger =
 LoggerFactory.getLogger(CreateKeyGroup.class);

    public static String createKeyGroup(CloudFrontClient cloudFrontClient, String
 publicKeyId) {
        String keyGroupId = cloudFrontClient.createKeyGroup(b ->
 b.keyGroupConfig(c -> c
                .items(publicKeyId)
                .name("JavaKeyGroup" + UUID.randomUUID())))
                .keyGroup().id();
        logger.info("KeyGroup created with ID: [{}]", keyGroupId);
        return keyGroupId;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 CreateKeyGroup。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreatePublicKey** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreatePublicKey。

CLI

AWS CLI

创建 CloudFront 公有密钥

以下示例通过在名为 pub-key-config.json 的 JSON 文件中提供参数来创建 CloudFront 公有密钥。必须先拥有 PEM 编码的公有密钥，然后才能使用此命令。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的创建 RSA 密钥对。

```
aws cloudfront create-public-key \
    --public-key-config file://pub-key-config.json
```

文件 pub-key-config.json 是当前文件夹中包含以下内容的 JSON 文档：请注意，公有密钥以 PEM 格式编码。

```
{
    "CallerReference": "cli-example",
    "Name": "ExampleKey",
    "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
    "Comment": "example public key"
}
```

输出：

```
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/public-key/
KDFB19YGCR002",
    "ETag": "E2QWRUHEXAMPLE",
    "PublicKey": {
        "Id": "KDFB19YGCR002",
```

```
        "CreatedTime": "2019-12-05T18:51:43.781Z",
        "PublicKeyConfig": {
            "CallerReference": "cli-example",
            "Name": "ExampleKey",
            "EncodedKey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxPMbCA2Ks0lnd7IR+3pw
\nwd3H/7jPGwj8bLUmore7bX+oeGpZ6QmLAe/1UOWcmZX2u70dYcSIzB1ofZtcn4cJ
\nenHBAzO3ohBY/L1tQGJfS2A+omnN6H16VZE1JCK8XSJyfze7MDLcUyHZETdxuvRb
\nA9X343/vMAuQPnhinFJ8Wdy8YBXSPpy7r95ylUQd9LfYTBzVZYG2tSesplcOkjM3\n2Uu
+oMWxQAw1NINnSLPinMVsutJy6ZqlV3McWNWe4T+STGtWhrPNqJEn45sIcCx4\nq
+kGZ2NQ0FyIyT2eiLKOX5Rgb/a36E/aMk4VoDsaenBQgG7WLTnstb9sr7MIhS6A\nrwIDAQAB\n-----
END PUBLIC KEY-----\n",
            "Comment": "example public key"
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 CreatePublicKey。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

以下代码示例读取公有密钥并将其上传到 Amazon CloudFront。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class CreatePublicKey {
```

```
    private static final Logger logger =
LoggerFactory.getLogger(CreatePublicKey.class);

    public static String createPublicKey(CloudFrontClient cloudFrontClient,
String publicKeyFileName) {
        try (InputStream is =
CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
            String publicKeyString = IoUtils.toUtf8String(is);
            CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
                    .createPublicKey(b -> b.publicKeyConfig(c -> c
                            .name("JavaCreatedPublicKey" + UUID.randomUUID())
                            .encodedKey(publicKeyString)
                            .callerReference(UUID.randomUUID().toString())));
            String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
            logger.info("Public key created with id: [{}]", createdPublicKeyId);
            return createdPublicKeyId;

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 CreatePublicKey。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteDistribution** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteDistribution。

CLI

AWS CLI

删除 CloudFront 分配

以下示例将删除 ID 为 EDFDVBD6EXAMPLE 的 CloudFront 分配。删除分配之前，必须先禁用它。要禁用分配，请使用 update-distribution 命令。有关更多信息，请参阅 update-distribution 示例。

分配已禁用，您可以将其删除。要删除分配，您必须使用 `--if-match` 选项来提供分配的 ETag。要获取 ETag，请使用 get-distribution 或 get-distribution-config 命令。

```
aws cloudfront delete-distribution \
    --id EDFDVBD6EXAMPLE \
    --if-match E2QWRUHEXAMPLE
```

成功时，此命令没有输出。

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 DeleteDistribution。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

以下代码示例将分配更新为禁用，使用 waiter 等待更改部署完成，然后删除该分配。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;

public class DeleteDistribution {
        private static final Logger logger =
 LoggerFactory.getLogger(DeleteDistribution.class);

        public static void deleteDistribution(final CloudFrontClient
 cloudFrontClient, final String distributionId) {
                // First, disable the distribution by updating it.
                GetDistributionResponse response =
 cloudFrontClient.getDistribution(b -> b
```

```
                                .id(distributionId));
                String etag = response.eTag();
                DistributionConfig distConfig =
response.distribution().distributionConfig();

                cloudFrontClient.updateDistribution(builder -> builder
                                .id(distributionId)
                                .distributionConfig(builder1 -> builder1

.cacheBehaviors(distConfig.cacheBehaviors())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())
                                                        .enabled(false)
                                                        .origins(distConfig.origins())
                                                        .comment(distConfig.comment())

.callerReference(distConfig.callerReference())

.defaultCacheBehavior(distConfig.defaultCacheBehavior())

.priceClass(distConfig.priceClass())
                                                        .aliases(distConfig.aliases())
                                                        .logging(distConfig.logging())

.defaultRootObject(distConfig.defaultRootObject())

.customErrorResponses(distConfig.customErrorResponses())

.httpVersion(distConfig.httpVersion())

.isIPV6Enabled(distConfig.isIPV6Enabled())

.restrictions(distConfig.restrictions())

.viewerCertificate(distConfig.viewerCertificate())
                                                        .webACLId(distConfig.webACLId())

.originGroups(distConfig.originGroups()))
                                .ifMatch(etag));

                logger.info("Distribution [{}] is DISABLED, waiting for
deployment before deleting ...",
                                distributionId);
                GetDistributionResponse distributionResponse;
```

```
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distributionId)).matched();
                        distributionResponse = responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Could not disable distribution"));
                }

                DeleteDistributionResponse deleteDistributionResponse =
 cloudFrontClient
                                .deleteDistribution(builder -> builder
                                                .id(distributionId)

 .ifMatch(distributionResponse.eTag()));
                if (deleteDistributionResponse.sdkHttpResponse().isSuccessful())
 {
                        logger.info("Distribution [{}] DELETED", distributionId);
                }
        }
}
```

- 有关 API 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 DeleteDistribution。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetCloudFrontOriginAccessIdentity** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetCloudFrontOriginAccessIdentity`。

CLI

AWS CLI

获取 CloudFront 源访问身份

以下示例获取 ID 为 E74FTE3AEXAMPLE 的 CloudFront 源访问身份（OAI），包括其 ETag 和关联的 S3 规范 ID。OAI ID 将在 create-cloud-front-origin-access-identity 和 list-cloud-front-origin-access-identities 命令的输出中返回。

```
aws cloudfront get-cloud-front-origin-access-identity --id E74FTE3AEXAMPLE
```

输出：

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentity": {
        "Id": "E74FTE3AEXAMPLE",
        "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
        "CloudFrontOriginAccessIdentityConfig": {
            "CallerReference": "cli-example",
            "Comment": "Example OAI"
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetCloudFrontOriginAccessIdentity。

PowerShell

适用于 PowerShell 的工具

示例 1：此示例返回由 -Id 参数指定的特定 Amazon CloudFront 源访问身份。尽管不需要 -Id 参数，但如果您不指定该参数，则不会返回任何结果。

```
Get-CFCloudFrontOriginAccessIdentity -Id E3XXXXXXXXXXRT
```

输出：

```
      CloudFrontOriginAccessIdentityConfig    Id
     S3CanonicalUserId
      -----------------------------------    --
     -----------------
```

```
     Amazon.CloudFront.Model.CloudFrontOr... E3XXXXXXXXXXRT
    4b6e...
```

- 有关 API 的详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetCloudFrontOriginAccessIdentity。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 GetCloudFrontOriginAccessIdentityConfig 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetCloudFrontOriginAccessIdentityConfig。

CLI

AWS CLI

获取 CloudFront 源访问身份配置

以下示例获取有关 ID 为 E74FTE3AEXAMPLE 的 CloudFront 源访问身份（OAI）的元数据，包括其 ETag。OAI ID 将在 create-cloud-front-origin-access-identity 和 list-cloud-front-origin-access-identities 命令的输出中返回。

```
aws cloudfront get-cloud-front-origin-access-identity-config --id E74FTE3AEXAMPLE
```

输出：

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "CloudFrontOriginAccessIdentityConfig": {
        "CallerReference": "cli-example",
        "Comment": "Example OAI"
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetCloudFrontOriginAccessIdentityConfig。

PowerShell

适用于 PowerShell 的工具

示例 1：此示例返回有关由 -Id 参数指定的单个 Amazon CloudFront 源访问身份的配置信息。如果未指定 -Id 参数，则会出现错误。

```
Get-CFCloudFrontOriginAccessIdentityConfig -Id E3XXXXXXXXXXRT
```

输出：

```
     CallerReference                                                Comment
     ---------------                                                -------
     mycallerreference: 2/1/2011 1:16:32 PM                         Caller
reference: 2/1/2011 1:16:32 PM
```

- 有关 API 的详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetCloudFrontOriginAccessIdentityConfig。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetDistribution** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetDistribution。

CLI

AWS CLI

获取 CloudFront 分配

以下示例获取有关 ID 为 EDFDVBD6EXAMPLE 的 CloudFront 分配，包括其 ETag。分配 ID 将在 create-distribution 和 list-distributions 命令中返回。

```
aws cloudfront get-distribution --id EDFDVBD6EXAMPLE
```

输出：

```
{
    "ETag": "E2QWRUHEXAMPLE",
```

```
    "Distribution": {
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "Deployed",
        "LastModifiedTime": "2019-12-04T23:35:41.433Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-example",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
```

```
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
```

```
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetDistribution。

PowerShell

适用于 PowerShell 的工具

示例 1：检索有关特定分配的信息。

```
Get-CFDistribution -Id EXAMPLE0000ID
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 GetDistribution。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetDistributionConfig** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetDistributionConfig。

CLI

AWS CLI

获取 CloudFront 分配配置

以下示例获取有关 ID 为 EDFDVBD6EXAMPLE 的 CloudFront 分配的元数据，包括其 ETag。分配 ID 将在 create-distribution 和 list-distributions 命令中返回。

```
aws cloudfront get-distribution-config --id EDFDVBD6EXAMPLE
```

输出：

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "DistributionConfig": {
        "CallerReference": "cli-example",
        "Aliases": {
            "Quantity": 0
        },
        "DefaultRootObject": "index.html",
        "Origins": {
            "Quantity": 1,
            "Items": [
                {
                    "Id": "awsexamplebucket.s3.amazonaws.com-cli-example",
                    "DomainName": "awsexamplebucket.s3.amazonaws.com",
                    "OriginPath": "",
                    "CustomHeaders": {
                        "Quantity": 0
                    },
                    "S3OriginConfig": {
                        "OriginAccessIdentity": ""
                    }
                }
            ]
        },
        "OriginGroups": {
            "Quantity": 0
```

```
        },
        "DefaultCacheBehavior": {
            "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-example",
            "ForwardedValues": {
                "QueryString": false,
                "Cookies": {
                    "Forward": "none"
                },
                "Headers": {
                    "Quantity": 0
                },
                "QueryStringCacheKeys": {
                    "Quantity": 0
                }
            },
            "TrustedSigners": {
                "Enabled": false,
                "Quantity": 0
            },
            "ViewerProtocolPolicy": "allow-all",
            "MinTTL": 0,
            "AllowedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
            "DefaultTTL": 86400,
            "MaxTTL": 31536000,
            "Compress": false,
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "FieldLevelEncryptionId": ""
        },
```

```
        "CacheBehaviors": {
            "Quantity": 0
        },
        "CustomErrorResponses": {
            "Quantity": 0
        },
        "Comment": "",
        "Logging": {
            "Enabled": false,
            "IncludeCookies": false,
            "Bucket": "",
            "Prefix": ""
        },
        "PriceClass": "PriceClass_All",
        "Enabled": true,
        "ViewerCertificate": {
            "CloudFrontDefaultCertificate": true,
            "MinimumProtocolVersion": "TLSv1",
            "CertificateSource": "cloudfront"
        },
        "Restrictions": {
            "GeoRestriction": {
                "RestrictionType": "none",
                "Quantity": 0
            }
        },
        "WebACLId": "",
        "HttpVersion": "http2",
        "IsIPV6Enabled": true
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 GetDistributionConfig。

PowerShell

适用于 PowerShell 的工具

示例 1：检索有关特定分配的配置。

```
Get-CFDistributionConfig -Id EXAMPLE0000ID
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell 命令参考》中的 GetDistributionConfig。

Python

SDK for Python（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
```

```
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 GetDistributionConfig。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `ListCloudFrontOriginAccessIdentities` 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListCloudFrontOriginAccessIdentities`。

CLI

AWS CLI

列出 CloudFront 源访问身份

以下示例获取您的 AWS 账户中的 CloudFront 源访问身份（OAI）列表：

```
aws cloudfront list-cloud-front-origin-access-identities
```

输出：

```
{
    "CloudFrontOriginAccessIdentityList": {
        "Items": [
            {
                "Id": "E74FTE3AEXAMPLE",
                "S3CanonicalUserId":
 "cd13868f797c227fbea2830611a26fe0a21ba1b826ab4bed9b7771c9aEXAMPLE",
```

```
                "Comment": "Example OAI"
            },
            {
                "Id": "EH1HDMBEXAMPLE",
                "S3CanonicalUserId":
 "1489f6f2e6faacaae7ff64c4c3e6956c24f78788abfc1718c3527c263bf7a17EXAMPLE",
                "Comment": "Test OAI"
            },
            {
                "Id": "E2X2C9TEXAMPLE",
                "S3CanonicalUserId":
 "cbfeebb915a64749f9be546a45b3fcfd3a31c779673c13c4dd460911ae402c2EXAMPLE",
                "Comment": "Example OAI #2"
            }
        ]
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》 中的 ListCloudFrontOriginAccessIdentities。

PowerShell

适用于 PowerShell 的工具

示例 1：此示例返回 Amazon CloudFront 源访问身份的列表。由于 -maxItem 参数指定的值为 2，因此结果包含两个身份。

```
Get-CFCloudFrontOriginAccessIdentityList -MaxItem 2
```

输出：

```
IsTruncated : True
Items       : {E326XXXXXXXXXT, E1YWXXXXXXX9B}
Marker      :
MaxItems    : 2
NextMarker  : E1YXXXXXXXXX9B
Quantity    : 2
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 ListCloudFrontOriginAccessIdentities。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListDistributions** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListDistributions。

CLI

AWS CLI

列出 CloudFront 分配

以下示例将获取您 AWS 账户中的 CloudFront 分配列表：

```
aws cloudfront list-distributions
```

输出：

```
{
    "DistributionList": {
        "Items": [
            {
                "Id": "EMLARXS9EXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
EMLARXS9EXAMPLE",
                "Status": "InProgress",
                "LastModifiedTime": "2019-11-22T00:55:15.705Z",
                "InProgressInvalidationBatches": 0,
                "DomainName": "d111111abcdef8.cloudfront.net",
                "ActiveTrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "DistributionConfig": {
                    "CallerReference": "cli-example",
                    "Aliases": {
                        "Quantity": 0
                    },
                    "DefaultRootObject": "index.html",
                    "Origins": {
                        "Quantity": 1,
                        "Items": [
```

```
                            {
                                "Id": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                                "DomainName":
 "awsexamplebucket.s3.amazonaws.com",
                                "OriginPath": "",
                                "CustomHeaders": {
                                    "Quantity": 0
                                },
                                "S3OriginConfig": {
                                    "OriginAccessIdentity": ""
                                }
                            }
                        ]
                    },
                    "OriginGroups": {
                        "Quantity": 0
                    },
                    "DefaultCacheBehavior": {
                        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-cli-
example",
                        "ForwardedValues": {
                            "QueryString": false,
                            "Cookies": {
                                "Forward": "none"
                            },
                            "Headers": {
                                "Quantity": 0
                            },
                            "QueryStringCacheKeys": {
                                "Quantity": 0
                            }
                        },
                        "TrustedSigners": {
                            "Enabled": false,
                            "Quantity": 0
                        },
                        "ViewerProtocolPolicy": "allow-all",
                        "MinTTL": 0,
                        "AllowedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
```

```
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
```

```
                    "WebACLId": "",
                    "HttpVersion": "http2",
                    "IsIPV6Enabled": true
                }
            },
            {
                "Id": "EDFDVBD6EXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
EDFDVBD6EXAMPLE",
                "Status": "InProgress",
                "LastModifiedTime": "2019-12-04T23:35:41.433Z",
                "InProgressInvalidationBatches": 0,
                "DomainName": "d930174dauwrn8.cloudfront.net",
                "ActiveTrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "DistributionConfig": {
                    "CallerReference": "cli-example",
                    "Aliases": {
                        "Quantity": 0
                    },
                    "DefaultRootObject": "index.html",
                    "Origins": {
                        "Quantity": 1,
                        "Items": [
                            {
                                "Id": "awsexamplebucket1.s3.amazonaws.com-cli-
example",
                                "DomainName":
 "awsexamplebucket1.s3.amazonaws.com",
                                "OriginPath": "",
                                "CustomHeaders": {
                                    "Quantity": 0
                                },
                                "S3OriginConfig": {
                                    "OriginAccessIdentity": ""
                                }
                            }
                        ]
                    },
                    "OriginGroups": {
                        "Quantity": 0
                    },
```

```
                    "DefaultCacheBehavior": {
                        "TargetOriginId": "awsexamplebucket1.s3.amazonaws.com-
cli-example",
                        "ForwardedValues": {
                            "QueryString": false,
                            "Cookies": {
                                "Forward": "none"
                            },
                            "Headers": {
                                "Quantity": 0
                            },
                            "QueryStringCacheKeys": {
                                "Quantity": 0
                            }
                        },
                        "TrustedSigners": {
                            "Enabled": false,
                            "Quantity": 0
                        },
                        "ViewerProtocolPolicy": "allow-all",
                        "MinTTL": 0,
                        "AllowedMethods": {
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ],
                            "CachedMethods": {
                                "Quantity": 2,
                                "Items": [
                                    "HEAD",
                                    "GET"
                                ]
                            }
                        },
                        "SmoothStreaming": false,
                        "DefaultTTL": 86400,
                        "MaxTTL": 31536000,
                        "Compress": false,
                        "LambdaFunctionAssociations": {
                            "Quantity": 0
                        },
                        "FieldLevelEncryptionId": ""
                    },
```

```
                    "CacheBehaviors": {
                        "Quantity": 0
                    },
                    "CustomErrorResponses": {
                        "Quantity": 0
                    },
                    "Comment": "",
                    "Logging": {
                        "Enabled": false,
                        "IncludeCookies": false,
                        "Bucket": "",
                        "Prefix": ""
                    },
                    "PriceClass": "PriceClass_All",
                    "Enabled": true,
                    "ViewerCertificate": {
                        "CloudFrontDefaultCertificate": true,
                        "MinimumProtocolVersion": "TLSv1",
                        "CertificateSource": "cloudfront"
                    },
                    "Restrictions": {
                        "GeoRestriction": {
                            "RestrictionType": "none",
                            "Quantity": 0
                        }
                    },
                    "WebACLId": "",
                    "HttpVersion": "http2",
                    "IsIPV6Enabled": true
                }
            },
            {
                "Id": "E1X5IZQEXAMPLE",
                "ARN": "arn:aws:cloudfront::123456789012:distribution/
E1X5IZQEXAMPLE",
                "Status": "Deployed",
                "LastModifiedTime": "2019-11-06T21:31:48.864Z",
                "DomainName": "d2e04y12345678.cloudfront.net",
                "Aliases": {
                    "Quantity": 0
                },
                "Origins": {
                    "Quantity": 1,
                    "Items": [
```

```
                        {
                            "Id": "awsexamplebucket2",
                            "DomainName": "awsexamplebucket2.s3.us-
west-2.amazonaws.com",
                            "OriginPath": "",
                            "CustomHeaders": {
                                "Quantity": 0
                            },
                            "S3OriginConfig": {
                                "OriginAccessIdentity": ""
                            }
                        }
                    ]
                },
                "OriginGroups": {
                    "Quantity": 0
                },
                "DefaultCacheBehavior": {
                    "TargetOriginId": "awsexamplebucket2",
                    "ForwardedValues": {
                        "QueryString": false,
                        "Cookies": {
                            "Forward": "none"
                        },
                        "Headers": {
                            "Quantity": 0
                        },
                        "QueryStringCacheKeys": {
                            "Quantity": 0
                        }
                    },
                    "TrustedSigners": {
                        "Enabled": false,
                        "Quantity": 0
                    },
                    "ViewerProtocolPolicy": "allow-all",
                    "MinTTL": 0,
                    "AllowedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ],
                        "CachedMethods": {
```

```
                            "Quantity": 2,
                            "Items": [
                                "HEAD",
                                "GET"
                            ]
                        }
                    },
                    "SmoothStreaming": false,
                    "DefaultTTL": 86400,
                    "MaxTTL": 31536000,
                    "Compress": false,
                    "LambdaFunctionAssociations": {
                        "Quantity": 0
                    },
                    "FieldLevelEncryptionId": ""
                },
                "CacheBehaviors": {
                    "Quantity": 0
                },
                "CustomErrorResponses": {
                    "Quantity": 0
                },
                "Comment": "",
                "PriceClass": "PriceClass_All",
                "Enabled": true,
                "ViewerCertificate": {
                    "CloudFrontDefaultCertificate": true,
                    "MinimumProtocolVersion": "TLSv1",
                    "CertificateSource": "cloudfront"
                },
                "Restrictions": {
                    "GeoRestriction": {
                        "RestrictionType": "none",
                        "Quantity": 0
                    }
                },
                "WebACLId": "",
                "HttpVersion": "HTTP1_1",
                "IsIPV6Enabled": true
            }
        ]
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 ListDistributions。

PowerShell

适用于 PowerShell 的工具

示例 1：返回分配。

```
Get-CFDistributionList
```

- 有关 API 详细信息，请参阅《AWS Tools for PowerShell Cmdlet 参考》中的 ListDistributions。

Python

SDK for Python（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def list_distributions(self):
        print("CloudFront distributions:\n")
        distributions = self.cloudfront_client.list_distributions()
        if distributions["DistributionList"]["Quantity"] > 0:
            for distribution in distributions["DistributionList"]["Items"]:
                print(f"Domain: {distribution['DomainName']}")
                print(f"Distribution Id: {distribution['Id']}")
```

```
                print(
                    f"Certificate Source: "
                    f"{distribution['ViewerCertificate']['CertificateSource']}"
                )
                if distribution["ViewerCertificate"]["CertificateSource"] ==
 "acm":
                    print(
                        f"Certificate: {distribution['ViewerCertificate']
['Certificate']}"
                    )
                print("")
        else:
            print("No CloudFront distributions detected.")
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 ListDistributions。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateDistribution** 与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 UpdateDistribution。

CLI

AWS CLI

更新 CloudFront 分配的默认根对象

以下示例将 ID 为 EDFDVBD6EXAMPLE 的 CloudFront 分配的默认根对象更新为 index.html：

```
aws cloudfront update-distribution --id EDFDVBD6EXAMPLE \
    --default-root-object index.html
```

输出：

```
{
    "ETag": "E2QWRUHEXAMPLE",
    "Distribution": {
```

```
        "Id": "EDFDVBD6EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EDFDVBD6EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:55:39.870Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "6b10378d-49be-4c4b-a642-419ccaf8f3b5",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "example-website",
                        "DomainName": "www.example.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "CustomOriginConfig": {
                            "HTTPPort": 80,
                            "HTTPSPort": 443,
                            "OriginProtocolPolicy": "match-viewer",
                            "OriginSslProtocols": {
                                "Quantity": 2,
                                "Items": [
                                    "SSLv3",
                                    "TLSv1"
                                ]
                            },
                            "OriginReadTimeout": 30,
                            "OriginKeepaliveTimeout": 5
                        }
                    }
                ]
            },
            "OriginGroups": {
```

```
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "example-website",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 1,
                        "Items": [
                            "*"
                        ]
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ],
                    "CachedMethods": {
                        "Quantity": 2,
                        "Items": [
                            "HEAD",
                            "GET"
                        ]
                    }
                },
                "SmoothStreaming": false,
                "DefaultTTL": 86400,
                "MaxTTL": 31536000,
                "Compress": false,
                "LambdaFunctionAssociations": {
```

```
                    "Quantity": 0
                },
                "FieldLevelEncryptionId": ""
            },
            "CacheBehaviors": {
                "Quantity": 0
            },
            "CustomErrorResponses": {
                "Quantity": 0
            },
            "Comment": "",
            "Logging": {
                "Enabled": false,
                "IncludeCookies": false,
                "Bucket": "",
                "Prefix": ""
            },
            "PriceClass": "PriceClass_All",
            "Enabled": true,
            "ViewerCertificate": {
                "CloudFrontDefaultCertificate": true,
                "MinimumProtocolVersion": "TLSv1",
                "CertificateSource": "cloudfront"
            },
            "Restrictions": {
                "GeoRestriction": {
                    "RestrictionType": "none",
                    "Quantity": 0
                }
            },
            "WebACLId": "",
            "HttpVersion": "http1.1",
            "IsIPV6Enabled": true
        }
    }
}
```

更新 CloudFront 分配

以下示例通过在名为 `dist-config-disable.json` 的 JSON 文件中提供分配配置来禁用 ID 为 `EMLARXS9EXAMPLE` 的 CloudFront 分配。要更新分配，您必须使用 `--if-match` 选项来提供分配的 `ETag`。要获取 `ETag`，请使用 get-distribution 或 get-distribution-config 命令。

使用以下示例禁用分配后，您可以使用 delete-distribute 命令将其删除。

```
aws cloudfront update-distribution \
    --id EMLARXS9EXAMPLE \
    --if-match E2QWRUHEXAMPLE \
    --distribution-config file://dist-config-disable.json
```

文件 dist-config-disable.json 是当前文件夹中包含以下内容的 JSON 文档：请注意，Enabled 字段设置为 false：

```
{
    "CallerReference": "cli-1574382155-496510",
    "Aliases": {
        "Quantity": 0
    },
    "DefaultRootObject": "index.html",
    "Origins": {
        "Quantity": 1,
        "Items": [
            {
                "Id": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
                "DomainName": "awsexamplebucket.s3.amazonaws.com",
                "OriginPath": "",
                "CustomHeaders": {
                    "Quantity": 0
                },
                "S3OriginConfig": {
                    "OriginAccessIdentity": ""
                }
            }
        ]
    },
    "OriginGroups": {
        "Quantity": 0
    },
    "DefaultCacheBehavior": {
        "TargetOriginId": "awsexamplebucket.s3.amazonaws.com-1574382155-273939",
        "ForwardedValues": {
            "QueryString": false,
            "Cookies": {
                "Forward": "none"
            },
            "Headers": {
```

```
                "Quantity": 0
            },
            "QueryStringCacheKeys": {
                "Quantity": 0
            }
        },
        "TrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "ViewerProtocolPolicy": "allow-all",
        "MinTTL": 0,
        "AllowedMethods": {
            "Quantity": 2,
            "Items": [
                "HEAD",
                "GET"
            ],
            "CachedMethods": {
                "Quantity": 2,
                "Items": [
                    "HEAD",
                    "GET"
                ]
            }
        },
        "SmoothStreaming": false,
        "DefaultTTL": 86400,
        "MaxTTL": 31536000,
        "Compress": false,
        "LambdaFunctionAssociations": {
            "Quantity": 0
        },
        "FieldLevelEncryptionId": ""
    },
    "CacheBehaviors": {
        "Quantity": 0
    },
    "CustomErrorResponses": {
        "Quantity": 0
    },
    "Comment": "",
    "Logging": {
        "Enabled": false,
```

```
        "IncludeCookies": false,
        "Bucket": "",
        "Prefix": ""
    },
    "PriceClass": "PriceClass_All",
    "Enabled": false,
    "ViewerCertificate": {
        "CloudFrontDefaultCertificate": true,
        "MinimumProtocolVersion": "TLSv1",
        "CertificateSource": "cloudfront"
    },
    "Restrictions": {
        "GeoRestriction": {
            "RestrictionType": "none",
            "Quantity": 0
        }
    },
    "WebACLId": "",
    "HttpVersion": "http2",
    "IsIPV6Enabled": true
}
```

输出：

```
{
    "ETag": "E9LHASXEXAMPLE",
    "Distribution": {
        "Id": "EMLARXS9EXAMPLE",
        "ARN": "arn:aws:cloudfront::123456789012:distribution/EMLARXS9EXAMPLE",
        "Status": "InProgress",
        "LastModifiedTime": "2019-12-06T18:32:35.553Z",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d111111abcdef8.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-1574382155-496510",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "index.html",
```

```
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id":
"awsexamplebucket.s3.amazonaws.com-1574382155-273939",
                        "DomainName": "awsexamplebucket.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        }
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId":
"awsexamplebucket.s3.amazonaws.com-1574382155-273939",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
                    },
                    "Headers": {
                        "Quantity": 0
                    },
                    "QueryStringCacheKeys": {
                        "Quantity": 0
                    }
                },
                "TrustedSigners": {
                    "Enabled": false,
                    "Quantity": 0
                },
                "ViewerProtocolPolicy": "allow-all",
                "MinTTL": 0,
                "AllowedMethods": {
                    "Quantity": 2,
                    "Items": [
```

```
                    "HEAD",
                    "GET"
                ],
                "CachedMethods": {
                    "Quantity": 2,
                    "Items": [
                        "HEAD",
                        "GET"
                    ]
                }
            },
            "SmoothStreaming": false,
            "DefaultTTL": 86400,
            "MaxTTL": 31536000,
            "Compress": false,
            "LambdaFunctionAssociations": {
                "Quantity": 0
            },
            "FieldLevelEncryptionId": ""
        },
        "CacheBehaviors": {
            "Quantity": 0
        },
        "CustomErrorResponses": {
            "Quantity": 0
        },
        "Comment": "",
        "Logging": {
            "Enabled": false,
            "IncludeCookies": false,
            "Bucket": "",
            "Prefix": ""
        },
        "PriceClass": "PriceClass_All",
        "Enabled": false,
        "ViewerCertificate": {
            "CloudFrontDefaultCertificate": true,
            "MinimumProtocolVersion": "TLSv1",
            "CertificateSource": "cloudfront"
        },
        "Restrictions": {
            "GeoRestriction": {
                "RestrictionType": "none",
                "Quantity": 0
```

```
                }
            },
            "WebACLId": "",
            "HttpVersion": "http2",
            "IsIPV6Enabled": true
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的 UpdateDistribution。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.DistributionConfig;
import
 software.amazon.awssdk.services.cloudfront.model.UpdateDistributionRequest;
import software.amazon.awssdk.services.cloudfront.model.CloudFrontException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ModifyDistribution {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <id>\s

                Where:
                    id - the id value of the distribution.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String id = args[0];
        CloudFrontClient cloudFrontClient = CloudFrontClient.builder()
                .region(Region.AWS_GLOBAL)
                .build();

        modDistribution(cloudFrontClient, id);
        cloudFrontClient.close();
    }

    public static void modDistribution(CloudFrontClient cloudFrontClient, String
idVal) {
        try {
            // Get the Distribution to modify.
            GetDistributionRequest disRequest = GetDistributionRequest.builder()
                    .id(idVal)
                    .build();

            GetDistributionResponse response =
cloudFrontClient.getDistribution(disRequest);
            Distribution disObject = response.distribution();
            DistributionConfig config = disObject.distributionConfig();

            // Create a new DistributionConfig object and add new values to
comment and
            // aliases
            DistributionConfig config1 = DistributionConfig.builder()
                    .aliases(config.aliases()) // You can pass in new values here
                    .comment("New Comment")
                    .cacheBehaviors(config.cacheBehaviors())
                    .priceClass(config.priceClass())
```

```
                    .defaultCacheBehavior(config.defaultCacheBehavior())
                    .enabled(config.enabled())
                    .callerReference(config.callerReference())
                    .logging(config.logging())
                    .originGroups(config.originGroups())
                    .origins(config.origins())
                    .restrictions(config.restrictions())
                    .defaultRootObject(config.defaultRootObject())
                    .webACLId(config.webACLId())
                    .httpVersion(config.httpVersion())
                    .viewerCertificate(config.viewerCertificate())
                    .customErrorResponses(config.customErrorResponses())
                    .build();

            UpdateDistributionRequest updateDistributionRequest =
 UpdateDistributionRequest.builder()
                    .distributionConfig(config1)
                    .id(disObject.id())
                    .ifMatch(response.eTag())
                    .build();

            cloudFrontClient.updateDistribution(updateDistributionRequest);

        } catch (CloudFrontException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 UpdateDistribution。

Python

SDK for Python（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
class CloudFrontWrapper:
    """Encapsulates Amazon CloudFront operations."""

    def __init__(self, cloudfront_client):
        """
        :param cloudfront_client: A Boto3 CloudFront client
        """
        self.cloudfront_client = cloudfront_client


    def update_distribution(self):
        distribution_id = input(
            "This script updates the comment for a CloudFront distribution.\n"
            "Enter a CloudFront distribution ID: "
        )

        distribution_config_response =
 self.cloudfront_client.get_distribution_config(
            Id=distribution_id
        )
        distribution_config = distribution_config_response["DistributionConfig"]
        distribution_etag = distribution_config_response["ETag"]

        distribution_config["Comment"] = input(
            f"\nThe current comment for distribution {distribution_id} is "
            f"'{distribution_config['Comment']}'.\n"
            f"Enter a new comment: "
        )
        self.cloudfront_client.update_distribution(
            DistributionConfig=distribution_config,
            Id=distribution_id,
            IfMatch=distribution_etag,
        )
        print("Done!")
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的 UpdateDistribution。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 的 CloudFront 场景

以下代码示例演示如何通过 AWS SDK 实施 CloudFront 中的常见场景。这些场景演示了如何通过在 CloudFront 中调用多个函数来完成特定任务。每个场景都包含一个指向 GitHub 的链接，其中包含了有关如何设置和运行代码的说明。

示例

- 使用 AWS SDK 删除 CloudFront 签名资源
- 使用 AWS SDK 创建签名网址和 Cookie

使用 AWS SDK 删除 CloudFront 签名资源

以下代码示例演示了如何删除用于访问 Amazon Simple Storage Service (Amazon S3) 桶中的受限内容的资源。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.DeleteKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.DeleteOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.DeletePublicKeyResponse;
import software.amazon.awssdk.services.cloudfront.model.GetKeyGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.GetOriginAccessControlResponse;
import software.amazon.awssdk.services.cloudfront.model.GetPublicKeyResponse;
```

```
public class DeleteSigningResources {
    private static final Logger logger =
 LoggerFactory.getLogger(DeleteSigningResources.class);

    public static void deleteOriginAccessControl(final CloudFrontClient
 cloudFrontClient,
            final String originAccessControlId) {
        GetOriginAccessControlResponse getResponse = cloudFrontClient
                .getOriginAccessControl(b -> b.id(originAccessControlId));
        DeleteOriginAccessControlResponse deleteResponse =
 cloudFrontClient.deleteOriginAccessControl(builder -> builder
                .id(originAccessControlId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Origin Access Control [{}]",
 originAccessControlId);
        }
    }

    public static void deleteKeyGroup(final CloudFrontClient cloudFrontClient,
 final String keyGroupId) {

        GetKeyGroupResponse getResponse = cloudFrontClient.getKeyGroup(b ->
 b.id(keyGroupId));
        DeleteKeyGroupResponse deleteResponse =
 cloudFrontClient.deleteKeyGroup(builder -> builder
                .id(keyGroupId)
                .ifMatch(getResponse.eTag()));
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Key Group [{}]", keyGroupId);
        }
    }

    public static void deletePublicKey(final CloudFrontClient cloudFrontClient,
 final String publicKeyId) {
        GetPublicKeyResponse getResponse = cloudFrontClient.getPublicKey(b ->
 b.id(publicKeyId));

        DeletePublicKeyResponse deleteResponse =
 cloudFrontClient.deletePublicKey(builder -> builder
                .id(publicKeyId)
                .ifMatch(getResponse.eTag()));
```

```
        if (deleteResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Successfully deleted Public Key [{}]", publicKeyId);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的以下主题。
  - DeleteKeyGroup
  - DeleteOriginAccessControl
  - DeletePublicKey

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS SDK 创建签名网址和 Cookie

以下代码示例演示了如何创建允许访问受限资源的签名 URL 和 Cookie。

Java

SDK for Java 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 AWS 代码示例存储库中进行设置和运行。

使用 CannedSignerRequest 类对具有标准 策略的 URL 或 Cookie 进行签名。

```
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCannedPolicyRequest {
```

```
    public static CannedSignerRequest createRequestForCannedPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expirationDate = Instant.now().plus(7, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CannedSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expirationDate)
                .build();
    }
}
```

使用 CustomSignerRequest 类对具有自定义 策略的 URL 或 Cookie 进行签名。`activeDate` 和 `ipRange` 是可选方法。

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
 distributionDomainName,
            String fileNameToUpload,
            String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;
```

```
        String cloudFrontUrl = new URL(protocol, distributionDomainName,
 resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
                .resourceUrl(cloudFrontUrl)
                .privateKey(path)
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

以下示例演示如何使用 CloudFrontUtilities 类生成签名 Cookie 和 URL。请访问 GitHub，查看此代码示例。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
 LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
 CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
 cannedSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
```

```
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
 customSignerRequest) {
        SignedUrl signedUrl =
 cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
 getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
 cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
 getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
 cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
 cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
 cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 CloudFrontUtilities。

有关 AWS SDK 开发人员指南和代码示例的完整列表，请参阅 将 CloudFront 与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

文档历史记录

下表介绍了对 CloudFront 文档进行的重要更改。要接收有关更新的通知，您可以订阅 RSS 源。

变更	说明	日期
添加了新的托管式缓存策略	添加了新的托管式缓存策略 `UseOriginCacheControlHeaders` 和 `UseOriginCacheControlHeaders-QueryStrings` 。	2024 年 5 月 24 日
添加了源访问控制支持	您现在可以为 AWS Elemental MediaPackage V2 和 AWS Lambda 函数 URL 创建源访问控制（OAC）。	2024 年 4 月 11 日
CMCD 的实时日志字段	添加了 18 个用于实时日志记录的常用媒体客户端数据（CMCD）字段。	2024 年 4 月 9 日
开始使用基本 CloudFront 分配	更新了使用 Amazon S3 源和源访问控制（OAC）的基本分配教程。	2024 年 3 月 18 日
使用 AWS SDK 的 CloudFront 代码示例	新增的代码示例演示如何将 CloudFront 与 AWS 软件开发工具包（SDK）一起使用。这些示例被划分成代码摘录，展示如何调用单个服务函数，以及展示如何通过在同一服务中调用多个函数来完成特定任务。	2024 年 2 月 16 日
AWS 托管策略更新	`CloudFrontReadOnlyAccess` 和 `CloudFrontFullAccess` IAM 策略现	2023 年 12 月 19 日

	在支持 KeyValueStore 操作。	
JavaScript 运行时 2.0	添加了适用于 CloudFront Functions 的 JavaScript 运行时 2.0 功能。	2023 年 11 月 21 日
CloudFront KeyValueStore	Amazon CloudFront 现在支持 CloudFront KeyValueStore。此功能是一个安全、全局、低延迟的键值数据存储服务，允许您从 CloudFront Functions 内部进行读取访问，从而在 CloudFront 边缘站点实现高级可自定义逻辑。	2023 年 11 月 21 日
Lambda@Edge 支持更新的运行时版本	Lambda@Edge 现在支持带有 Node.js 20 运行时系统的 Lambda 函数。	2023 年 11 月 15 日
安全控制面板	当您创建分配时，CloudFront 会创建一个安全控制面板。启用 AWS WAF，管理地理限制，并查看请求、机器人和日志的概要数据。	2023 年 11 月 8 日
对函数中的查询字符串进行排序	CloudFront 现在支持使用 CloudFront Functions 对查询字符串进行排序。	2023 年 10 月 3 日
AWS WAF安全建议	Amazon CloudFront 现在在 CloudFront 控制台上显示AWS WAF安全建议。	2023 年 9 月 26 日
支持提供过时（过期）的缓存内容	CloudFront 支持 Stale-While-Revalidate 和 Stale-If-Error 缓存控制指令。	2023 年 5 月 15 日

一键启用 AWS WAF 保护	一种向 CloudFront 分配添加 AWS WAF 安全保护的简化方法。	2023 年 5 月 10 日
为用于标准日志的新 S3 存储桶启用 ACL	添加了说明和链接，以解决新 S3 存储桶的默认 ACL 设置。	2023 年 4 月 11 日
使用 Amazon S3 对象 Lambda 创建源	您可以使用 Amazon S3 对象 Lambda 接入点别名作为您分配的源。	2023 年 3 月 31 日
使用 CloudFront Functions 自定义 HTTP 状态和正文	您可以使用 CloudFront Functions 更新查看器响应状态代码并替换或删除响应正文。	2023 年 3 月 29 日
添加了 CORS 标头端口通配符选项	现在，您可以在 CORS 访问控制标头中包含端口的通配符配置。	2023 年 3 月 20 日
为《AWS Security Hub 用户指南》添加了新链接	更新了语言，并在《AWS Security Hub 用户指南》中添加了指向重组后的 Amazon CloudFront 控件的链接。	2023 年 3 月 9 日
CloudFront 现在在源请求策略中支持阻止列表（“除以下范围之外的所有”）	在源请求策略中使用阻止列表，以在 CloudFront 向源发送的请求中包含所有查询字符串、HTTP 标头或 Cookie，除了 指定的那些值。	2023 年 2 月 22 日
CloudFront 添加了新的托管源请求策略，用于转发除 Host 标头之外的所有查看器标头	使用 CloudFront 的新托管源请求策略将来自查看器请求的所有标头（除了 `Host` 标头）都包含在 CloudFront 向源发送的请求中。	2023 年 2 月 22 日

更新对 Lambda@Edge 的限制	Lambda@Edge 支持设置为自动的 Lambda 运行时管理配置。	2023 年 2 月 16 日
更新了 CloudFront 的 IAM 指南	更新了指南，使其符合 IAM 最佳实践。有关更多信息，请参阅 IAM 安全最佳实践。	2023 年 2 月 15 日
利用源访问控制增强安全性	现在，您可以通过仅允许访问指定的 CloudFront 分配来保护 MediaStore 源。	2023 年 2 月 9 日
用于确定查看器的标头结构的新标头	现在，您可以添加标头顺序和标头计数，以帮助根据查看器发送的标头来识别查看器。	2023 年 1 月 13 日
Lambda@Edge 支持更新的运行时版本	Lambda@Edge 现在对于 Node.js 18 运行时系统支持 Lambda 函数。	2023 年 1 月 12 日
使用响应标头策略来删除响应标头	现在，您可以使用 CloudFront 响应标头策略来删除 CloudFront 在来自源的响应中收到的标头。指定的标头不会包括在 CloudFront 发送给查看器的响应中。	2023 年 1 月 3 日
用于安全地测试配置更改的持续部署	现在，您可以通过测试一部分生产流量来部署对 CDN 配置的更改。	2022 年 11 月 18 日
发布 `CloudFront-Viewer-JA3-Fingerprint` 标头	现在，您可以使用 JA3 指纹来帮助确定请求是否来自已知客户端。	2022 年 11 月 16 日
添加了 CORS 标头通配符选项	现在，您可以在某些 CORS 访问控制标头中使用各种通配符配置。	2022 年 11 月 11 日

CloudFront 分配的其他指标	支持 CloudFront API 和 AWS CloudFormation 中的 `MonitoringSubscription` 。	2022 年 10 月 3 日
利用源访问控制增强安全性	现在，您可以通过仅允许访问指定的 CloudFront 分配来保护 Amazon S3 源。	2022 年 8 月 24 日
对 CloudFront 分配的 HTTP/3 支持	现在，可以为您的 CloudFront 分配选择 HTTP/3。	2022 年 8 月 15 日
将握手详细信息添加到 CloudFront-Viewer-TLS 标头	您可以重新查看有关所使用的 SSL/TLS 握手的信息。	2022 年 6 月 27 日
Server-Timing 标头中的新指标	已将新的 `cdn-downstream-fbl` 指标添加到 `Server-Timing` 标头。	2022 年 6 月 13 日
用于获取有关 TLS 版本和密码的信息的新标头	现在，您可以使用 `CloudFront-Viewer-TLS` 标头，以获取有关 TLS（或 SSL）版本以及用于查看器和 CloudFront 之间连接的密码的信息。	2022 年 5 月 23 日
适用于 CloudFront Functions 的新 FunctionThrottles 指标	利用 Amazon CloudWatch，您现在可以监控 CloudFront 函数在给定时间段内受到限制的次数。	2022 年 5 月 4 日
CloudFront 支持 Lambda 函数 URL	如果您使用带有函数 URL 的 Lambda 函数构建无服务器 Web 应用程序，则现在可以添加 CloudFront 以获得一系列好处。	2022 年 4 月 6 日

HTTP 响应中的 Server-Timing 标头	现在，您可以在从 CloudFront 发送的 HTTP 响应中启用 `Server-Timing` 标头，以查看可以帮助您深入了解 CloudFront 的行为和性能的指标。	2022 年 3 月 30 日
使用 AWS 托管的前缀列表来限制入站流量	现在，您可以限制仅从属于 CloudFront 面向源的服务器的 IP 地址传入您的源的 HTTP 和 HTTPS 流量。	2022 年 2 月 7 日
新功能	CloudFront 增加了对响应标头策略的支持，从而让您能够指定 CloudFront 将在其发送给查看器（Web 浏览器或其他客户端）的 HTTP 响应中添加的 HTTP 标头。您可以指定所需的标头（及其值），而无需对源进行任何更改或编写任何代码。有关更多信息，请参阅在 CloudFront 响应中添加或删除 HTTP 标头。	2021 年 11 月 2 日
新的 `CloudFront-Viewer-Address` 请求标头	CloudFront 增加了对新标头 `CloudFront-Viewer-Address` 的支持，该标头包含向 CloudFront 发送 HTTP 请求的查看器的 IP 地址。有关更多信息，请参阅添加 CloudFront 请求标头。	2021 年 10 月 25 日
Lambda@Edge 支持更新的运行时版本	Lambda@Edge 现在支持运行时版本为 Python 3.9 的 Lambda 函数。有关更多信息，请参阅支持的运行时。	2021 年 9 月 22 日

AWS 托管策略更新	CloudFront 更新了 CloudFrontReadOnlyAccess 策略。有关更多信息，请参阅 CloudFront 对 AWS 托管式策略做出的更新。	2021 年 9 月 8 日
新功能	CloudFront 现在支持适用于 HTTPS 查看器连接的 ECDSA 证书。有关更多信息，请参阅查看器和 CloudFront 之间支持的协议和密码以及在 CloudFront 中使用 SSL/TLS 证书的要求。	2021 年 7 月 14 日
新功能	CloudFront 现在支持更多方式来将备用域名从一个分配移动到另一个分配，且无需联系 AWS Support。有关更多信息，请参阅将备用域名移动到其他分配。	2021 年 7 月 7 日
新安全策略	CloudFront 现在支持新安全策略 TLSv1.2_2021，并提供了一组较少的受支持的密码。有关更多信息，请参阅查看器和 CloudFront 之间支持的协议和密码。	2021 年 6 月 23 日
新功能	Amazon CloudFront 现在支持 CloudFront Functions，这是一种 CloudFront 的原生功能，您可以在 JavaScript 中编写轻量级函数，以实现大规模、延迟敏感的 CDN 自定义。有关更多信息，请参阅使用 CloudFront Functions 在边缘进行自定义。	2021 年 5 月 3 日

Lambda@Edge 支持更新的运行时版本	Lambda@Edge 现在支持运行时版本为 Node.js 14 的 Lambda 函数。有关更多信息，请参阅支持的运行时。	2021 年 4 月 29 日
删除 RTMP 分配的相关文档	已于 2020 年 12 月 31 日弃用 Amazon CloudFront 实时消息协议 (RTMP) 分配。RTMP 分配的文档现已从《Amazon CloudFront 开发人员指南》中删除。	2021 年 2 月 10 日
新定价选项	Amazon CloudFront 推出了 CloudFront Security Savings Bundle。它是一种简单的方法，可以在 AWS 账单中最高节省 30% 的 CloudFront 费用。有关更多信息，请参阅 Savings Bundle 常见问题解答。	2021 年 2 月 5 日
新教程	《Amazon CloudFront 开发人员指南》包括关于使用 Amazon CloudFront 在 Elastic Load Balancing 中限制对应用程序负载均衡器的访问的新教程。有关更多信息，请参阅限制访问应用程序负载均衡器。	2020 年 12 月 18 日
用于公有密钥管理的新选项	CloudFront 现在支持通过 CloudFront 控制台和 API 对签名 URL 和签名 Cookie 进行公有密钥管理，而无需访问 AWS 账户根用户。有关更多信息，请参阅指定可以创建签名 URL 和签名 Cookie 的签署人。	2020 年 10 月 22 日

新功能 – Origin Shield	CloudFront 现在支持 CloudFront Origin Shield，这是 CloudFront 缓存基础设施中的一个附加层，有助于最大限度减少源的负载、提高其可用性并降低其运营成本。有关更多信息，请参阅使用 Amazon CloudFront Origin Shield。	2020 年 10 月 20 日
新的压缩格式	现在，在将 CloudFront 配置为在 CloudFront 边缘站点压缩对象时，CloudFront 支持 Brotli 压缩格式。您也可以将 CloudFront 配置为使用标准化的 `Accept-Encoding` 标头来缓存 Brotli 对象。有关更多信息，请参阅提供压缩文件和压缩支持。	2020 年 9 月 14 日
新 TLS 协议	CloudFront 现在支持将 TLS 1.3 协议用于查看器和 CloudFront 分配之间的 HTTPS 连接。默认情况下，在所有 CloudFront 安全策略中启用 TLS 1.3。有关更多信息，请参阅查看器和 CloudFront 之间支持的协议和密码。	2020 年 9 月 3 日
新实时日志	CloudFront 现在支持可配置的实时日志。利用实时日志，您可以实时获取有关向分配发出的请求的信息。您可以使用实时日志来进行监控和分析，并根据内容交付性能采取相应措施。有关更多信息，请参阅实时日志。	2020 年 8 月 31 日

对其他指标的 API 支持	CloudFront 现在支持使用 CloudFront API 实现八个额外的实时指标。有关更多信息，请参阅启用其他指标。	2020 年 8 月 28 日
新的 CloudFront HTTP 标头	CloudFront 添加了额外的 HTTP 标头来确定查看器的相关信息，例如设备类型、地理位置等。有关更多信息，请参阅添加 CloudFront 请求标头。	2020 年 7 月 23 日
新功能	CloudFront 现在支持缓存策略和源请求策略，让您能够更精细地控制 CloudFront 分配的缓存键和源请求。有关更多信息，请参阅控制缓存键和控制源请求。	2020 年 7 月 22 日
新安全策略	CloudFront 现在支持新安全策略 TLSv1.2_2019，并提供了一组较少的受支持的密码。有关更多信息，请参阅查看器和 CloudFront 之间支持的协议和密码。	2020 年 7 月 8 日
新增了用于控制源超时和尝试次数的设置	CloudFront 添加了新设置来控制源超时和尝试次数。有关更多信息，请参阅控制源超时和尝试次数。	2020 年 6 月 5 日
新增了有关通过创建安全静态网站开始使用 CloudFront 的文档	通过使用 Amazon S3、CloudFront、Lambda@Edge 等（全部通过 AWS CloudFormation 进行部署）创建安全静态网站来开始使用 CloudFront。有关更多信息，请参阅安全静态网站入门。	2020 年 6 月 2 日

Lambda@Edge 支持更新的运行时版本	Lambda@Edge 现在支持运行时版本为 Node.js 12 和 Python 3.8 的 Lambda 函数。有关更多信息，请参阅支持的运行时。	2020 年 2 月 27 日
CloudWatch 中的新实时指标	现在，Amazon CloudFront 在 Amazon CloudWatch 中额外提供了 8 个实时指标。有关更多信息，请参阅启用其他 CloudFront 分配指标。	2019 年 12 月 19 日
访问日志中的新字段	CloudFront 为访问日志添加了七个新字段。有关更多信息，请参阅标准日志文件字段。	2019 年 12 月 12 日
AWS WordPress 插件	您可以使用 AWS WordPress 插件为访问您的 WordPress 网站的访客提供使用 CloudFront 的加速查看体验。（更新：自 2022 年 9 月 30 日起，AWS WordPress 插件将被弃用。）	2019 年 10 月 30 日
基于标签的 IAM 权限策略和资源级 IAM 权限策略	CloudFront 现在支持另外两种指定 IAM 权限策略的方法：基于标签的策略权限和资源级策略权限。有关更多信息，请参阅管理对资源的访问。	2019 年 8 月 8 日
支持 Python 编程语言	除了 Node.js，您现在还可以使用 Python 编程语言在 Lambda@Edge 中开发函数。有关涵盖各种场景的函数示例，请参阅 Lambda@Edge 函数示例。	2019 年 8 月 1 日

更新了监控图表	更新了内容以描述新的方法，让您可以直接从 CloudFront 控制台监控与 CloudFront 分配关联的 Lambda 函数，从而更轻松地跟踪和调试错误。有关更多信息，请参阅监控 CloudFront。	2019 年 6 月 20 日
整合了安全内容	新的“安全性”一章整合了有关 CloudFront 功能以及实施数据保护、IAM、日志记录、合规性等信息。关更多信息，请参阅安全性。	2019 年 5 月 24 日
现在需要验证域	CloudFront 现在要求使用 SSL 证书来验证您有权将备用域名用于分配。有关更多信息，请参阅使用备用域名和 HTTPS。	2019 年 4 月 9 日
更新了 PDF 文件名	《Amazon CloudFront 开发人员指南》的新文件名是：AmazonCloudFront_DevGuide。以前的名称为：cf-dg。	2019 年 1 月 7 日
新功能	CloudFront 现在支持 WebSocket，后者是一种基于 TCP 的协议，它在客户端和服务器之间需要长期连接时很有用。对于需要高可用性的场景，您现在还可以使用源故障转移功能设置 CloudFront。有关更多信息，请参阅将 WebSocket 与 CloudFront 分配结合使用和使用 CloudFront 源故障转移优化高可用性。	2018 年 11 月 20 日

新功能	CloudFront 现在支持运行 Lambda 函数的 HTTP 请求的详细错误日志记录。您可以将日志存储在 CloudWatch 中，并可以在您的函数返回无效响应时使用这些日志来帮助排查 HTTP 5xx 错误。有关更多信息，请参阅 Lambda 函数的 CloudWatch 指标和 CloudWatch Logs。	2018 年 10 月 8 日
新功能	您现在可以选择让 Lambda@Edge 为可写的 HTTP 方法（POST、PUT、DELETE 等）公开请求中的正文，以便您可以在 Lambda 函数中访问正文。您可以选择只读访问，也可以指定将替换正文。有关更多信息，请参阅选择“包含正文”选项以访问请求正文。	2018 年 8 月 14 日
新功能	CloudFront 现在支持使用 brotli 或其他压缩算法（包括或代替 gzip）压缩的内容。有关更多信息，请参阅提供压缩文件。	2018 年 7 月 25 日
重新组织	《Amazon CloudFront 开发人员指南》已重新组织，简化了相关内容的查找并改进了可浏览性和导航。	2018 年 6 月 28 日

新功能	现在，Lambda@Edge 可让您访问面向源的事件内其他标头（包括自定义标头），从而支持您进一步自定义存储在 Amazon S3 存储桶中的内容的交付。有关更多信息，请参阅显示基于查看器位置和查看器设备类型的内容个性化的以下示例。	2018 年 3 月 20 日
新功能	现在，您可以使用 Amazon CloudFront，协商使用椭圆曲线数字签名算法 (ECDSA) 的 HTTPS 源连接。ECDSA 使用更小的密钥，这种密钥的速度更快，但与旧的 RSA 算法一样安全。有关更多信息，请参阅 CloudFront 和源之间的通信支持的 SSL/TLS 协议和密码和关于 RSA 和 ECDSA 密码。	2018 年 3 月 15 日
新功能	借助 Lambda@Edge，您能够执行 Lambda 函数以响应 Amazon CloudFront 从您的源接收的 HTTP 错误，从而能够从您的源自定义错误响应。有关更多信息，请参阅显示重定向到另一个位置以及生成具有 200 状态代码（正常）的响应的这些示例。	2017 年 12 月 21 日

新功能	全新 CloudFront 功能“字段级加密”有助于进一步增强敏感数据（如信用卡号码或社会保险号码之类的个人身份信息（PII））的安全性。有关更多信息，请参阅使用字段级加密帮助保护敏感数据。	2017 年 12 月 14 日
文档历史记录已存档	存档了较旧的文档历史记录。	2017 年 12 月 1 日