持续部署单页应用的7大技巧

OneAPM官方技术博客 2016-05-23 4748 访问 前端工具

【编者按】本文作者为 Parker Selbert，主要介绍高效部署静态托管应用的7大技巧，助你实现持续、安全且高效的交付流程。本文系国内 ITOM 管理平台 OneAPM 编译呈现。

单页应用不仅能提供非常丰富的用户体验，而且为持续开发开辟了一个完全不同于以前的新途径。将前端应用从服务器分离，从而简化团队职责的划分，是非常合理的策略。维护一个单独的前端代码库，允许团队通过 API ，快速迭代应用特性、改善交互功能。

然而，交付静态资源的过程并不都是如此顺利的。在开始持续部署静态资源之前，你的团队必须注意托管和交付中的陷阱。 以下是一些有关高效部署静态托管应用的技巧，帮助你实现持续、安全，以及最重要的，高效的交付流程。

1. 使用最先进的打包和部署工具

如果你的团队已经决定单独部署客户端和服务器代码，很有可能的是，服务器并不是使用Node.js语言编写的，但这并不能阻止你使用 Node.js 和 NPM 来构建和管理应用程序！你可以自由使用最先进的打包和开发工具，而不用管服务器端采用了什么框架。

一旦你的构建和测试过程不受服务器框架限制，也就释放了交付过程。一旦前端应用通过了集成测试，CI 服务器就可以构建一个正式版(参见技巧2)，直接交付并进行发布(参见技巧5)。

2. 缩小，压缩和源映射(Source Maps)是必不可少的

部署一个单页应用远不止上传级联码到服务器这么简单。你在为生产环境的 Web 框架部署资源时，一定会精打细算地节省字节数，部署单页应用也是如此，需要同等的注意力与投入。这意味着单页应用必须尽可能缩小，压缩，并包含 源映射(source maps)。

任何主流的 JavaScript 构建工具，再加上少量的脚本，都能帮助你交付出最优化的应用。

3. 优化代码和样式交付

鉴于最近的趋势是将视图组件与样式定义放在一起，这一点可能稍有争议。但是，你需要权衡样式和代码捆绑后的利弊。

通常，浏览器可以并行下载 CSS 和 JS 文件，降低第一次加载后的绘制时间。如果所有的资源都捆绑在一起，是不可能提升性能的。当所有的样式和代码都捆绑在一个大文件内，客户只能盯着空白的屏幕，等待资源下载。

虽然多个文件会使交付过程稍显复杂，但文件缩小后带来的性能优势是值得我们这样做的。

4. 单独交付资源包

除非你是个极端的纯粹主义者，每个打包应用都应该是由库模块和应用代码组成的。通常，你的应用代码比库模块更改得更为频繁。当你提供巨大的级联包时，客户端被迫下载每一次更新，哪怕改动很小。应用程序包通常推送3MB 的数据量，这又需要下载大量的代码，而仅仅是因为几行应用代码的更改。

为了避免这个问题，你应该将应用程序分成至少两个资源包：一个包含级联库代码，另一个包含应用代码。未来如果实现 支持连接并行的HTTP / 2 协议，单个文件可以并行发送，这样的部署就不再必要了。但是现在，资源包的分割将加快用户获得每个新发布版本的速度。

5. 善加利用内容分发网络（CDN）

使用内容分发网络发布静态应用。只要保留语义缓存，CDN 就允许客户端继续指向相同的 URL。此外，在发布新代码时，即使缺乏资源指纹，也支持执行主动失效。主动失效会更新每个边缘服务器（也就是向客户端发布应用的服务器）上缓存的应用版本。

要注意的是，主动失效可能延时，在 Amazon CloudFront 上需要 10分钟或更多时间。这一不可预知的异步行为，是发布版本时需要额外留意的。

6. 连续性面前没有版本

不要期望用户会重新加载浏览器。假设一些用户会运行旧版本的应用，并做好准备，处理一些已弃用功能的请求。将版本发布看作是一个连续的变化，并决定你的发布周期。

总会某一阶段，继续支持所有旧版本及它们可能包含的各种错误，是不切实际的。除非你部署的是一台自助服务机，更新周期非常不频繁，你可以放心地假设用户会每周重新加载一次。

7. 逐步推出功能

使用功能标记逐步推出新功能。Ember 技术就是一个很好的例子，可以将功能和代码相绑定，但它是默认禁用的。代码在运行时动态产生，但是大多数人并不使用它。一旦通过测试人员或一小部分用户的测试，你就可以发布包含这一功能的新版本。

在发布服务器端代码时，通常也会使用同样的方法，但是静态托管单页应用的风险更高。循序渐进的方法是至关重要的，因为回滚代码的速度取决于 CDN 的失效期。这意味着你若是发布了一个错误版本，它至少在生产环境中运行10分钟以上，而无法立即撤销。

应用资源和服务器代码若是绑定，部署单页应用就变得既简单又稳定。此外，你可以利用原生JavaScript 工具的优势，而不管应用框架是什么。核心是，服务器/浏览器的关系是一个简单的分布式系统。通过在服务器端单独部署单页面应用，你的团队可以获得微系统架构带来的灵活性，专注度以及优先度。

原文地址：http://blog.codeship.com/continuously-deploying-single-page-apps/