



下载APP



19 | 实战痛点5：如何打包发布你的Vue 3应用？

2021-12-01 大圣

《玩转Vue 3全家桶》

[课程介绍 >](#)**讲述：大圣**

时长 08:07 大小 7.45M



你好，我是大圣。

在实战痛点 4 这一讲中，我们一起学习了 Vue 3 项目的性能优化策略。今天，我们来聊一下项目上线前的最后一步，就是如何把开发好的代码部署到线上。

对于这个问题，你可能脱口而出：“使用npm run build就好了呀”。这样做只是在本地把代码打包，如果想要在线上也可以访问这些代码，那么还需要加上部署的过程。所以在下面，我先给你介绍一下当前这个时代的前端代码在部署的时候，有哪些难点和问题需要处理。



代码部署难点

在 jQuery 时代之前，前端项目中所有的内容都是一些简单的静态资源。那个时候，网站还没有部署的概念，网站上线前，我们直接把开发完的项目打包发给运维，再由运维把代码直接上传到服务器的网站根目录下解压缩，这样就完成了项目的部署。

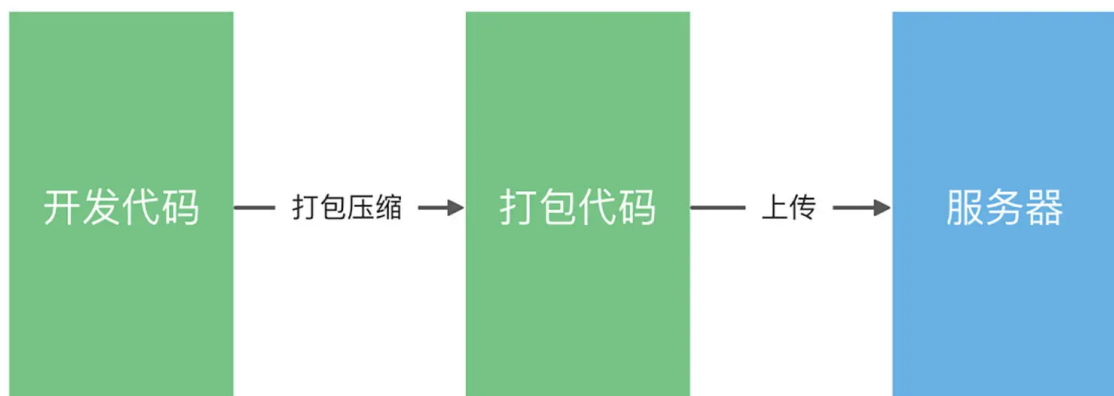
后来的 jQuery 时代，项目的入口页面被后端管理，模板部署到了后端，CSS、JavaScript 和图片等静态资源依然是打包到后端之后，再解压处理。但现在，我们对前端的性能和稳定性的要求也越来越高，jQuery 时代的那种简单的部署模式就不足以应对性能优化、持续部署等一系列的情境。

现在前端所处的时代，我们主要会面临后面这些代码部署难点：首先是，如何高效地利用项目中的文件缓存；然后是，如何能够让整个项目的上线部署过程自动化，尽可能避免人力的介入，从而提高上线的稳定性；最后，项目上线之后，如果发现有重大 Bug，我们就要考虑如何尽快回滚代码。

当我们面对这些代码部署上的难点，特别是在团队协作的项目中遇到时，我们就可以考虑对项目进行自动化部署了，这样代码部署的速度和稳定性会给项目研发效率带来很好的提升。

项目上线前的自动化部署

下图所示的，是大部分团队部署项目时的逻辑。实际上，大部分前端开发者都会认为，完成图示中的打包压缩这一步，也就是开发完项目之后，代码推送到 GitHub 后，就算完成任务了。但是，打包代码之后，把代码上传服务器也是这一步，对于前端开发者来说，是很少能接触到，但却是很重要的一步。

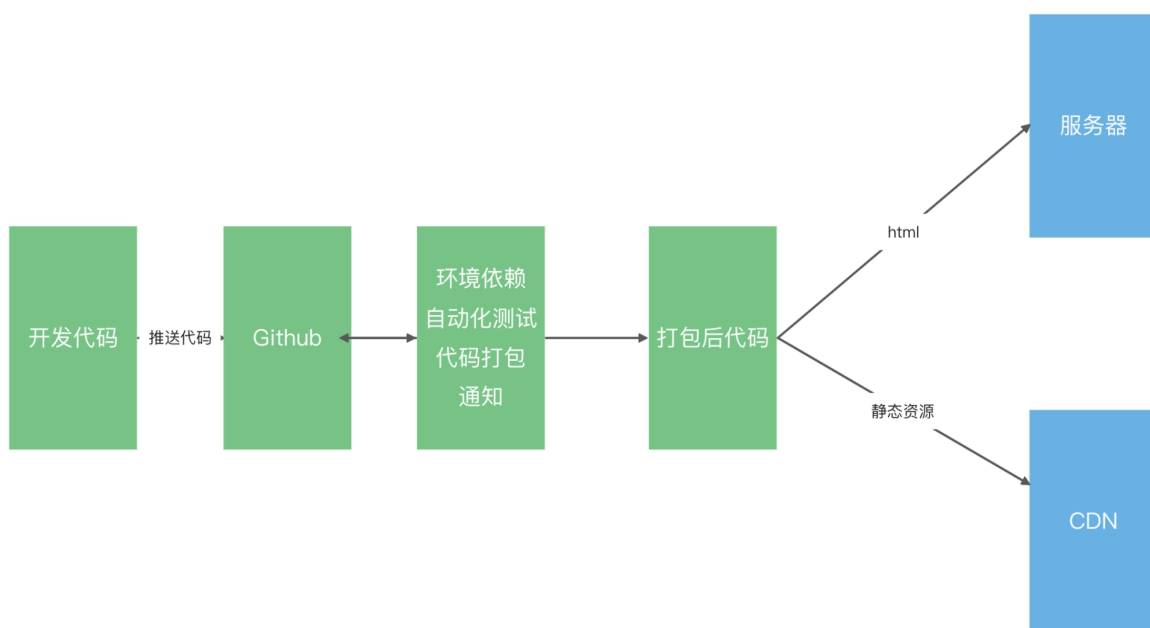


所以，对于如何把打包好的代码上传到服务器这个问题，就值得我们去好好探究，琢磨出一个好的解决方案。

首先，我们需要一台独立的机器去进行打包和构建的操作，这台机器需要独立于所有开发环境，这样做是为了保证打包环境的稳定；之后，在部署任务启动的时候，我们需要拉取远程的代码，并且切换到需要部署的分支，然后锁定 Node 版本进行依赖安装、单元测试、ESLint 等代码检查工作；最后，在这台机器上，执行经过编译产出的打包后的代码，并打包上传代码到 CDN 和静态服务器。当然了，完成这些操作之后，还要能通过脚本自动通过内部沟通软件通知团队项目构建的结果。

但是在项目部署的过程中，迎面而来的可能是下面这些问题：在什么操作系统环境中执行项目的构建？由谁触发构建？如何管理前面所述的把代码上传 CDN 时，CDN 账户的权限？如何自动化执行部署的全过程，如果每次都由人工执行，就得消耗一个人力守着编译打包了，而且较为容易引发问题，比如测试的步骤遗漏或部署顺序出错。那么如何提升构建速率，就成了部署功能中需要解决的重要问题。

为了解决上面这些问题，业界提出了一些解决方案：比如，采用能保证环境一致性的 Docker；自动化构建触发可以通过 GitHub Actions；GitHub 的 actions 功能相当于给我们提供了一个免费的服务器，可以很方便地监控代码的推送、安装依赖、代码编译自动上传到服务器。



上图所展示的，就是我们使用了 GitHub Actions 部署项目之后的项目开发流程。现在静态资源管理已经完成，也实现了自动化部署。提交代码之后，我们的项目就可以自动推送到服务器，这样，网站的第一次上线也就算成功了。

项目上线后的自动化部署

前端项目的自动化部署完成后，我们可以保证上线的稳定性，但是后续的持续上线怎么办？直接发到生产环境，会面临极大的风险。但如果不直接发布到生产环境，我们就不能在本地和测试的前端环境去连接生产环境的数据库。

所以我们需要一个**预发布的（Pre）环境**，这个环境只能让测试和开发人员访问，除了访问地址的环节不同，其他所有环节都和生产环境保持一致，从而提供最真实的回归测试环境。

这个时候，我们会遇见下面这些问题，首先，如果我们确定项目下个版本在下周一零点发布，那我们就只能晚上 12 点准时守在电脑前，等待结果吗？如果 npm 安装依赖失败，或者上线后发现了重大 Bug，那就只能迎接用户的吐槽吗？


其次，**随着 node_modules 的体积越来越大，构建时间会越来越长**。如果每次构建都需要 30 分钟甚至更长时间的话，那么，即使 Bug 是在项目刚上线时就发现的，并且你也秒级响应，并修复了 Bug，但在重新部署项目时，我们也需要等服务器慢慢编译。这个时候，时间就是金钱，如果你在修复 Bug 和重新部署项目上，耗费了过多的时间，那么就会导致项目故障时间过长的问題。

为了解决上面说到的这些问题，我们需要一种机制，能够让我们在发现问题之后，尽快地将版本进行回滚，并且在回滚的操作过程中，尽可能不需要人力的介入。所以，我们需要静态资源的版本管理，具体来说，就是让每个历史版本的资源都能保留下来，并且有一个唯一的版本号，如果发生了故障，能够瞬间切换版本。这个过程由具体的代码实现之后，我们只需要点击回滚的版本号，系统就会自动恢复到上线前的版本。

在这种机制下，如果你的业务流量特别大，每秒都有大量用户访问和使用，那么直接全量上线的操作就会被禁止。为了减少上线时，部署操作对用户造成的影响，我们需要先选择一部分用户去做灰度测试，也就是说，上线后的项目的访问权限，暂时只对这些用户开放。或者，你也可以做一些 AB 测试，比如给北京的同学推送 Vue 课，给上海的同学推荐

React 课等等。我们需要做的，就是把不同版本的代码分开打包，互不干涉。之后，我们再设计部署的机器和机房去适配不同的用户。

在 Gtihub 中，我们可以使用 actions 去配置打包的功能，下面的代码是 actions 的配置文件。在这个配置文件中，我们使用 Ubuntu 作为服务器的打包环境，然后拉取 GitHub 中最新的 master 分支代码，并且把 Node 版本固定为 14.7.6，执行 npm install 安装代码所需依赖后，再执行 npm run build 进行代码打包压缩。在下面的代码中，我们就通过 GitHub Actions 自动化打包了一份准备上线的代码。

 复制代码

```
1 name: 打包应用的actions
2 on:
3   push: # 监听代码时间
4     branches:
5       - master # master分支代码推送的时候激活当前action
6 jobs:
7   build:
8     # runs-on 操作系统
9     runs-on: ubuntu-latest
10    steps:
11      - name: 迁出代码
12        uses: actions/checkout@master
13      # 安装Node
14      - name: 安装Node
15        uses: actions/setup-node@v1
16        with:
17          node-version: 14.7.6
18      # 安装依赖
19      - name: 安装依赖
20        run: npm install
21      # 打包
22      - name: 打包
23        run: npm run build
24
```

然后，我们需要配置上线服务器和 GitHub Actions 服务器的信任关系，通过 SSH 密钥可以实现免登录直接部署。我们直接把 build 之后的代码打包压缩，通过 SSH 直接上传到服务器上，并且要进行代码文件版本的管理，就完成了代码的部署。

最后一步，就是部署成功后的结果通知了。现在办公软件钉钉和飞书都提供了相关的推送结果，我们可以随时通过群机器人接口把消息推送到群内，关于钉钉机器人的适用文档，

你直接看官方的 [🔗 开发文档](#) 就可以了，我们需要做的是把版本号、部署日期、发起人等信息推送到对应接口，这样就完成了自动化部署的操作。

这一过程涉及服务器、钉钉开发文档、GitHub Actions，浏览器和本地代码环境多个场景的转换，这一讲我们先重点学习整体部署需要的思路和注意事项，实际的部署操作过程后续我会录成视频来进行实操演示。

总结

今天的主要内容就讲完了，我们来总结一下今天学到的内容吧。首先，我们讲解了前端部署这一过程的难点，包括怎么处理缓存、怎么自动化部署等等。在部署上，我们需要尽可能减少人力的参与，做到整个过程都用代码可控。

之后，在前端自动化部署这一部分，我们着重讲到了代码的打包上传和项目的部署，其中需要你重点注意的是项目的部署。为了解决如何部署代码到线上这一问题，我们需要一个独立的部署系统，有了独立的部署系统之后，我们可以把整个部署上线的过程自动化。借助 GitHub 的 Actions，我们可以很方便地使用 actions 自带的服务去进行发布环节的版本确认、依赖安装、代码打包和上传的工作。

思考题

最后留一个思考题吧，你现在负责的项目中，发布和部署这个流程里有哪些环节可以优化呢？欢迎在评论区留言讨论，我们下一讲再见！

分享给需要的人，Ta订阅后你可得 **20** 元现金奖励

 生成海报并分享

 赞 10  提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 [加餐01 | 什么是好的项目？](#)

下一篇 [加餐02 | 深入TypeScript](#)

更多课程推荐

跟月影学可视化

系统掌握图形学与可视化核心原理

月影

奇虎 360 奇舞团团长

可视化 UI 框架 SpriteJS 核心开发者



新版升级：点击「👤请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

精选留言

写留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。