链滴

手把手教你如何巧用 Github 的 Action 功能

作者：xuexiangjys
原文链接：https://ld246.com/article/1610109838489
来源网站：链滴

概念

GitHub Actions 是 GitHub 于2018年10月推出的持续集成服务。

那么何谓持续集成呢?

持续集成

持续集成（Continuous integration），也就是我们经常说的CI。它是一种软件开发实践，可以让团在持续的基础上收到反馈并进行改进，不必等到开发后期才寻找和修复缺陷，常运用于软件的敏捷开中。Jenkins就是我们常用的持续集成平台工具。

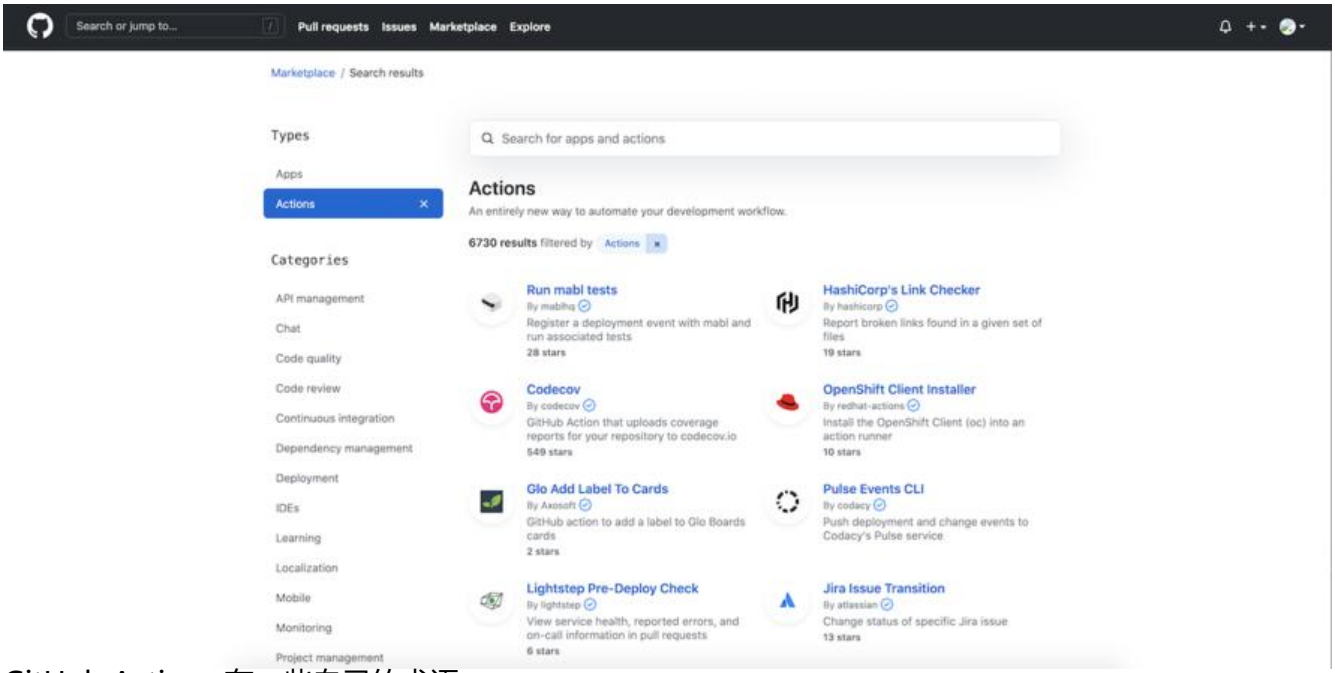
理解了持续集成的概念之后，下面我简单讲一下使用持续集成的好处：

- 提高效率，减少了重复性工作：一些重复性的工作写成脚本交给持续集成服务执行。
- 减少了人工带来的错误：机器通过预先写好的脚本执行犯错的几率比人工低很多。
- 减少等待的时间：一套完备的持续集成服务涵盖了开发、集成、测试、部署等各个环节。
- 提高产品质量：很多大公司在代码提交后都会有一套代码检视脚本（俗称门禁）来检查代码的提交否符合规范，从而从源头遏制问题的产生。

Actions

相比较持续集成这个大概念，GitHub推出的 Actions 就显得非常轻量和巧妙了。Actions就相当于持集成中的某个特定功能的脚本，通过多个actions的自由组合，便可实现自己特定功能的持续集成服
。

同时，Github为了方便大家使用 Actions，还专门做了一个 Actions市场, 真的是非常方便!



GitHub Actions 有一些自己的术语:

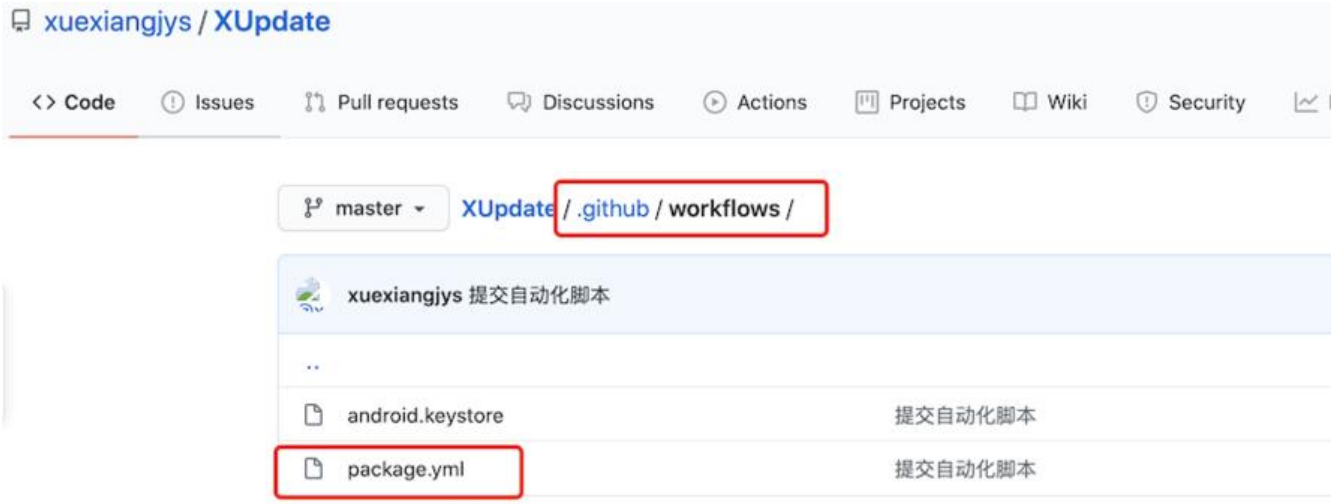
- 1. workflow（工作流程）：持续集成一次运行的过程，就是一个workflow。

原文链接：手把手教你如何巧用 Github 的 Action 功能

- 2. job（任务）：一个workflow由一个或多个jobs构成，含义是一次持续集成的运行，可以完成个任务。
- 3. step（步骤）：每个job由多个step构成，一步步完成。
- 4. action（动作）：每个step可以依次执行一个或多个命令（action）。

workflow文件

GitHub Actions 的配置文件叫做 workflow文件，存放在代码仓库的 .github/workflows目录, 如下所示：



workflow文件采用 YAML格式，文件名可以任意取，但是后缀名统一为 .yml，比如上图的 package.ml。

workflow文件的配置字段非常多，详见官方文档 。下面是一些基本字段：

- 1. name: workflow的名称。如果省略该字段，默认为当前workflow的文件名。
- 2. on: 触发workflow的条件，通常是某些事件，例如：release、push、pull_request等。详细容可以参照官方文档 。
- 3. jobs: workflow文件的主体内容，表示要执行的一项或多项任务。
  - jobs.<job_id>.name:job_id是任务的id，name是任务的描述。
  - jobs.<job_id>.runs-on:runs-on运行所需要的虚拟机环境,它是必填字段。
  - jobs.<job_id>.needs:needs指定当前任务的依赖关系，即运行顺序。
  - jobs.<job_id>.steps:steps指定每个任务的运行步骤，可以包含一个或多个步骤。

Actions的应用

如何使用Action发布flutter插件

之前我写过一篇《Flutter Plugin插件开发填坑指南》，讲的就是如何开发一个flutter插件并进行发

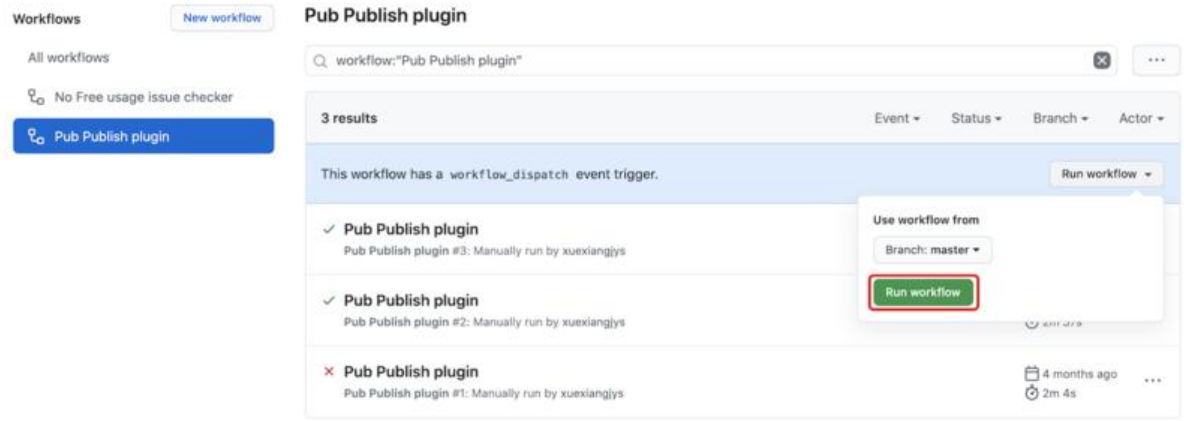
。但由于我们发布插件到 flutter插件平台 需要访问外网，而且还需要给命令终端设置代理，所以每的发布都非常的麻烦。

一个偶然的机会，我就在Action市场中发现了一个 publish-dart-flutter-package插件，可以一键把己的插件发布到 flutter插件平台 ，脚本如下:

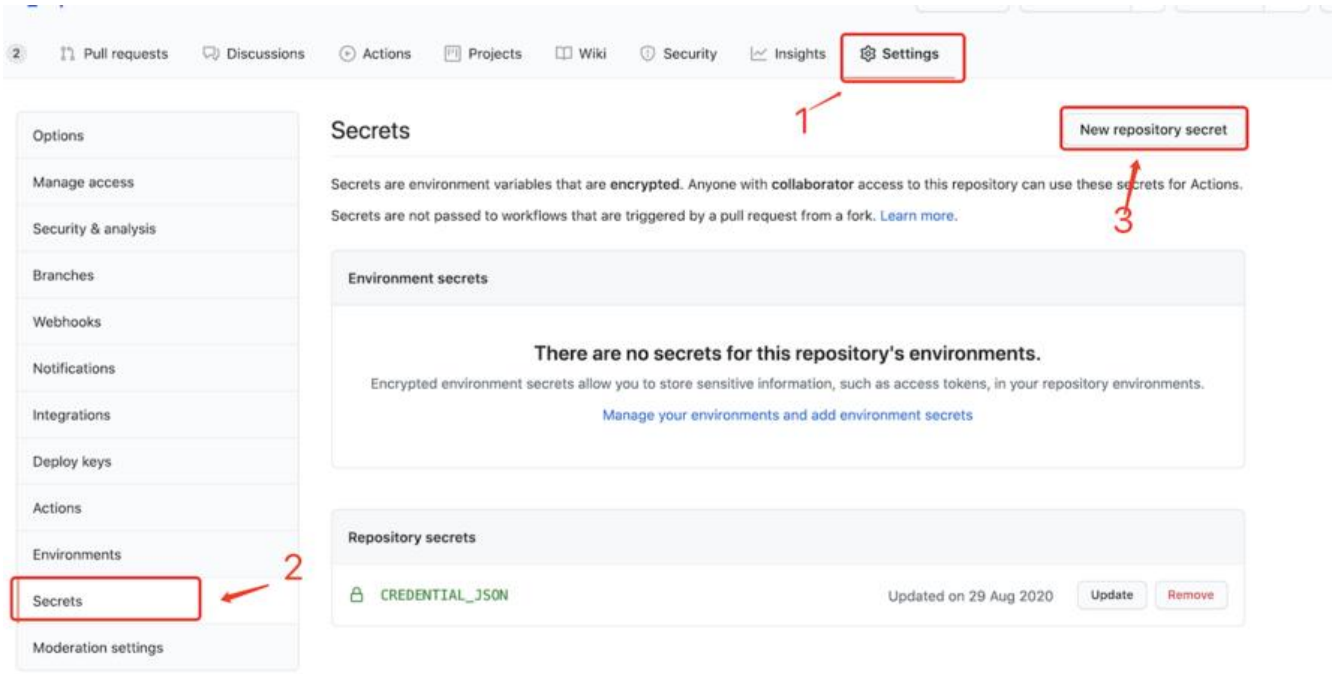
```
name: Pub Publish plugin

on: workflow_dispatch

jobs:
  publish:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v1
      - name: Publish
        uses: sakebook/actions-flutter-pub-publisher@v1.3.0
        with:
          credential: ${{ secrets.CREDENTIAL_JSON }}
          flutter_package: true
          skip_test: true
          dry_run: false
```

当然你也可以参考我的flutter_xupdate ，它就是利用这个Action进行发布的。

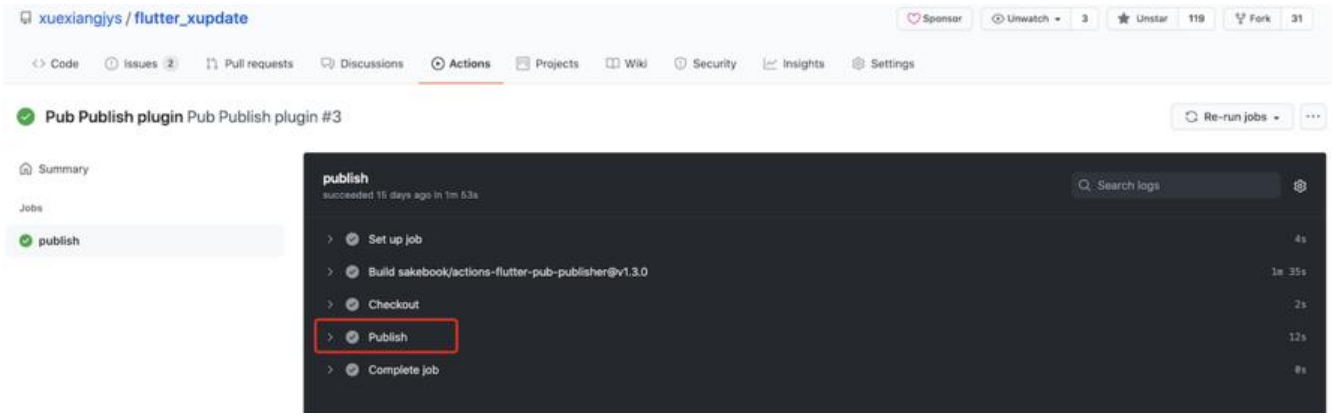
这里我们可以看到，我们定义的触发条件是 workflow_dispatch，也就是手动触发任务执行：需要我点击 Run workflow => 选择 master 分支 => 点击 Run workflow，入下图所示:



这里我们注意到定义了一个 secrets.CREDENTIAL_JSON常量，也就是我们的google账号认证证书，里需要我们在项目的 Settings => 选择 Secrets => 点击 New repository secret来创建一个属性名为REDENTIAL_JSON的常量。对应的值你可以到你的用户Home目录下的 .pub-cache文件夹下找到 credentials.json文件。



下图是我执行了一次发布action的结果，只需2分钟，无需科学上网工具和给命令终端配置代理，即完成flutter插件的发布，真的是非常方便!



如何使用Action打包apk

作为一名Android开发，你有没有想过每次提交代码或者发布版本的时候，github能够对应帮你自动包出一个apk?

这样你既可以省去打包apk的时间，还免去了应用包的管理，岂不美哉?

那么我们应该怎么做呢？下面就是我实现的一个 workflow脚本，主要的功能就是：在提交代码或者布版本的时候，自动构建脚本打包出apk，同时直接上传至 Artifacts存储。

```
name: Android CI

on:
  release:
    types: [published]
  push:
    branches:
      - master
    tags:
```

原文链接：手把手教你如何巧用 Github 的 Action 功能

```
      - '2.*'
  pull_request:
    branches:
      - master

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: set up JDK 1.8
        uses: actions/setup-java@v1
        with:
          java-version: 1.8
      - name: release apk sign
        run: |
          echo "给apk增加签名"
          cp $GITHUB_WORKSPACE/.github/workflows/android.keystore $GITHUB_WORKSPACE/
pp/android.keystore
          sed '$a\RELEASE_STORE_FILE=./android.keystore'          $GITHUB_WORKSPACE/gradle.p
operties -i
      - name: build with gradle
        run: |
          echo "开始进行release构建"
          chmod +x gradlew
          ./gradlew app:assembleRelease
      - name : upload apk
        uses: actions/upload-artifact@master
        if: always()
        with:
          name: xupdate_apk
          path: ${{ github.workspace }}/app/build/outputs/apk/release
```

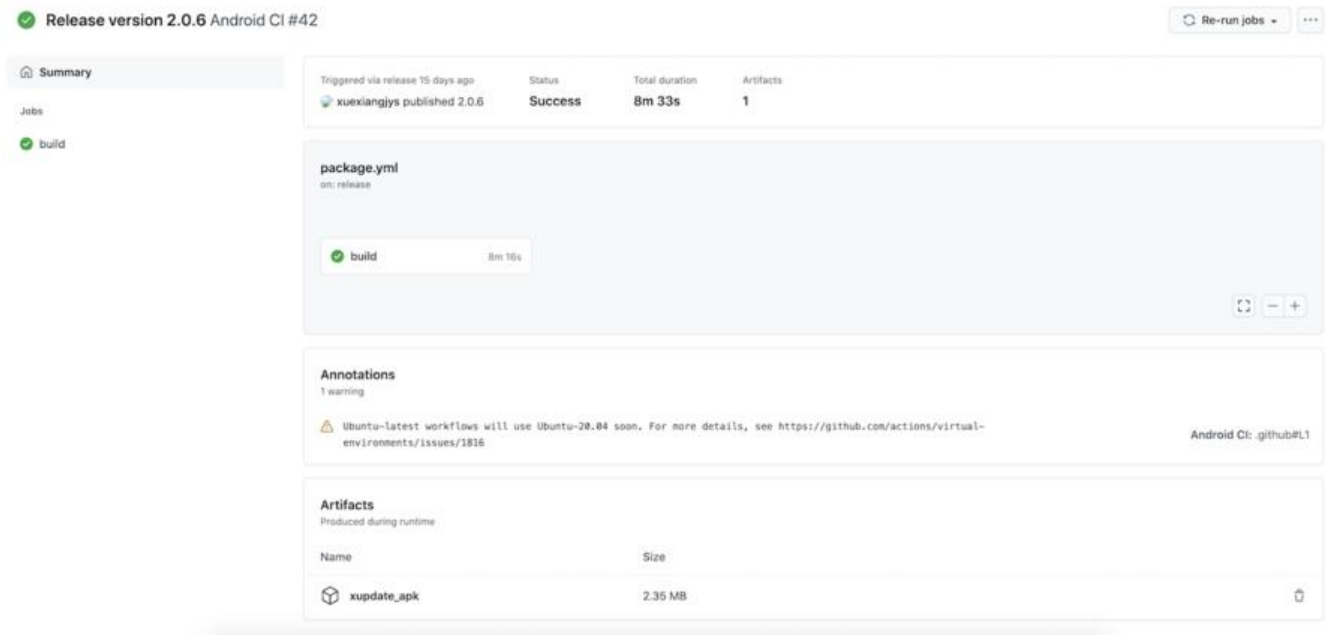
详细配置可以参考我的XUpdate 中的配置。

这里我们可以看到，我们定义的触发条件是 release，push和 pull_request，触发的分支是 master，ags是 2.*开头的。

整个任务主要分为4个步骤：

- 1. set up JDK 1.8： 构建java1.8的环境。
- 2. release apk sign：配置应用的签名。这里需要注意的是，这个地方的签名配置还是需要结合着uild.gradle 文件的配置来编写的。
- 3. build with gradle：编译构建apk。运行assembleRelease命令打release包。
- 4. upload apk：上传apk至Artifacts。

最后执行的效果如下：



如何使用Action来反击白嫖党

我在做开源项目的时候，经常能够碰到一些个无名小号（白嫖党），项目看都不看就提一些没有任何值的 issues，然后你好心好意地回复了，他却消失不见了...真的是让人恨得牙痒痒的！

是的，你没有看错，Action居然还可以用来反击白嫖党！这也是之前我在逛掘金的时候偶然看到一篇章《x 对白嫖怪 SAY NO !!! —— 如何在 GitHub 上阻止无耻白嫖》 发现的。

那么他是怎么做的呢？其实也很简单，就是设置触发的条件是 issues的创建，在创建的时候去查询一 issues的创建者是否 star或者 fork了该仓库，如果满足条件则不做处理，否则将自动锁住并关闭 issus。

当然，这位作者也是把这个非常骚的Action做成了一个插件，插件的地址是：https://github.com/marketplace/actions/no-free-usage-action ，使用起来非常简单。

以下是我简单使用的脚本案例：

```
name: No Free usage issue checker

on:
  issues:
    types: [opened, reopened]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - name: Check issue actor
        uses: fluttercandies/no-free-usage-action@v1.0.1
        with:
          token: ${{ secrets.GITHUB_TOKEN }} # 由GitHub提供的临时Token，必须在此处进行传递且必须为这个值。
          forked: '--no-forked'
```

原文链接：手把手教你如何巧用 Github 的 Action 功能

words: To support our project, please file the issue after you starred the repo. Thanks! :lightly_smiling_face:

这里，我设置的触发条件是 issues的打开和重新打开事件，设置不强制 fork，但是需要 star。当一野生的白嫖党出没并在你的项目上提 issues的时候，就会触发下图的效果：

看到上图的效果，是不是感到很惊喜，很刺激？你以为你做白嫖党我就没有办法治你？哈哈，给我老点！

最后

都看到这儿了，还不赶紧三连支持一下，难道你也想做白嫖党吗？

更多资讯内容，欢迎搜索我的微信公众号:【我的Android开源之旅】