Edwardwu Lv2
2020年02月28日 阅读 299

githubactions进行github仓库和gitee仓库同步

为何要做同步

`github`是国际通用的代码托管平台，奈何身处国内，往往受限于网速，如果**个人博客**，**项目主页**托管在`github`上的话，访问速度也是十分堪忧

让我们把目光瞄准国内，国内的知名替代品主要是`gitee`和`coding`，因为在国内，同时有`pages`的这两家成了我们的首选，这里以`gitee`为例子,让两个`github`和`gitee`的仓库和`pages`自动进行同步

为何可以同步

`git`设计之初其实就是分布式管理的，同步很正常

同步方案

同步方案主要三种

通过 `remote` 设定不同的别名，上传两遍

gitee =》 github

这应该是最常用的一种， `git` 远程仓库选用国内的 `gitee` ,然后通过 `gitee` 的 webhook 触发 `devops` 进行同步，缺点是要自己准备 `devops` ，出一份钱

github =》 gitee

借助万能的 `github actions` 在代码**提交**,**合并**之后自动调用写好的 `action` 进行同步

当然，人工点也可以， `gitee` 提供了对应服务

github => gitee 同步方案

代码库同步

借助 `wearerequired/git-mirror-action@master` 这个 `git` 自动同步的 `actions` , 我们可以实现任意仓库自动同步
闲着没事上去点一下也没问题

actions

以下是我使用的 `.github/workflows/sync.yml` 记得修改 **source-repo** 和 **destination-repo**为对应的两个仓库

```yml
name: Sync To Gitee
on: page_build
jobs:
  sync:
    runs-on: ubuntu-latest
    steps:
      - uses: wearerequired/git-mirror-action@master
        env:
          SSH_PRIVATE_KEY: ${{ secrets.GITEE_PRIVATE_KEY }}
        with:
          source-repo: "git@gitee.com:mizuka/Mizuka.gitee.io.git"
          destination-repo: "git@gitee.com:mizuka/Mizuka.gitee.io.git"
```

对应的触发条件

添加钥匙

接下来是增加**私钥**, 因为同步过程通过 `ssh` 触发，我们需要使用 `ssh-keygen` 命令生成一对**公钥**和**私钥**，注意命名，然后将**公钥**（***.pub）的内容添加到 `github` 和 `gitee` 的可信名单里

接下来，在对应的仓库 `setting` 的 `secrets` 中添加**GITEE_PRIVATE_KEY**,内容为之前的**私匙**

这样子，每次部署之后， `github` 会自动推送到 `gitee` 上

pages 同步

如何开启 pages

- **github** 普通项目 `gh-pages` 分支和 `用户名.github.io` 的 `master` 分支会自动开启 `github pages`
- **gitee** 服务里有 `gtiee pages` 服务,点击打开

gitee 的问题

`gitee` 的 `pages` 服务如果没有花费**99/年**开启 `gitee pages pro` 服务的话，每次仓库有更新的话，是不会自动更新重新部署的，也就是说你还得进去**更新部署一次**

gitee 自动部署方案

花钱

99/年,有钱可以为所欲为

人工

上去自己点一下重新部署就行啦

模拟登录调用

其实抓取以下接口就能发现 直接通过 `POST: https://gitee.com/${repository}/pages/rebuild` 这个接口进行的，那么我们只要模拟登录就可以了

```
axios.post(
  url + "/rebuild",
  qs.stringify({
    branch: core.getInput("branch"),
    build_directory: core.getInput("directory"),
    force_https: core.getInput("https")
  }),
  {
    headers: {
      "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",
      Cookie: cookie,
      Referer: url,
      "X-Requested-With": "XMLHttpRequest",
      "X-CSRF-Token": csrfToken
    }
  }
);
```

主要是 `headers` 里的 `Content-Type`，`X-CSRF-Token`，`Cookie` 特别重要，页面的参数对应这页面的几个按钮，应该很好理解

其中，`Cookie` 可以 `devtools` 中直接拿到，`X-CSRF-TOKEN` 经过调查，被写在了 `<meta content="" name="csrf-token"></meta>` 上，通过获取任意一个页面然后拿到即可，否则会**403**

具体参考我封装的插件

github actions

我个人封装了一个 `actions` 只要在 `steps` 上追加一个 `step` 即可

```yml
- name: reload
  uses: mizuka-wu/gitee-pages-action@v1.0.0
  with:
    repository: mizuka/Mizuka
    cookie: ${{ secrets.GITEE_COOKIE }}
```

当然，需要增加一个 `GITEE_COOKIE` 和设定 `repository`

查看文档