部署更轻松了，Github Action自动化部署Hexo：代码推送，云服务器自动部署

程序员哇子 2024-03-19 209 阅读5分钟

部署更轻松了，Github Action自动化部署：代码推送，云服务器自动部署

> 我的博客都部署在阿里云的服务器上，利用nginx做web服务，很麻烦的是，每次我在本地编写完代码，我都要手动在本地打包，再手动把打包后的文件传到服务器上，这个过程重复且枯燥，繁琐。可以说我简直是受够了！！于是我在群里受大佬的指导，发现了Github Action，能够自动化的完成打包，测试，部署等工作，我心里想：这简直太好了啊！那不得研究研究

关于Github Action

官网：docs.github.com/zh/actions/...

GitHub Actions 是一种持续集成和持续交付 (CI/CD) 平台，可用于自动执行生成、测试和部署管道。 您可以创建工作流程来构建和测试存储库的每个拉取请求，或将合并的拉取请求部署到生产环境。

GitHub Actions 不仅仅是 DevOps，还允许您在存储库中发生其他事件时运行工作流程。 例如，您可以运行工作流程，以便在有人在您的存储库中创建新问题时自动添加相应的标签。

GitHub 提供 Linux、Windows 和 macOS 虚拟机来运行工作流程，或者您可以在自己的数据中心或云基础架构中托管自己的自托管运行器。

简单来说：就是它是Github上类似于持续集成的功能，它允许你在一些节点上（push、tag）等特定时间触发一些操作，我们这里可以利用它实现自动部署应用到自己的服务器

没有使用Github Action，我需要

- 编辑好文章后，运行 `hexo clean` 清空缓存
- 运行 `hexo g` 生成html文件
- 如果使用的是github page服务，需要运行 `hexo deploy`，如果是自己的服务器，还要连上服务器上传打包好的文件

程序员哇子 LV.2
前端工程师 | 喜欢骑行
24 文章 5.8k 阅读 16 粉丝
关注 私信

相关推荐
工程提速，自动部署
213阅读 · 0点赞
hexo-CI自动部署
526阅读 · 1点赞
如何使用 GitHub Actions 自动部署 Hex...
7.7k阅读 · 32点赞
使用 GitHub Actions 自动部署 Hexo 博...
2.9k阅读 · 5点赞
打造Github Issue到Hexo部署自动工作...
1.4k阅读 · 5点赞

精选内容
Ai Agent 自研项目 VS 字节扣子，差点...
小傅哥 · 62阅读 · 0点赞
面试官：集群模式下，如何解决本地缓...
托尼学长 · 91阅读 · 2点赞
从零构建高并发锁工具：基于AOP与Ree...
Derek_Smart · 51阅读 · 2点赞
JDK 17 实战系列（第6期）：平台支持与...
用户8491371754... · 40阅读 · 0点赞
JustAuth实战系列（第6期）：策略模式...

使用Github Action，我需要

- 提交代码，github自动帮我们打包部署到服务器

> 内心os：太方便了 我要学我要学！！！

开始设置

1、去Github的自己仓库点击Actions，新建一个workflow工作流，应该会有模版，随便选择一个，反正yml的内容可以改，我们由于是node项目，可以选择node模版

参考我的博客的workflow：

yml 体验AI代码助手 代码解读 复制代码

```yml
# This workflow will do a clean installation of node dependencies, cache/restore them, build the sou
# For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-

name: hexo_deploy

on:
  push:
    branches: [ "main" ]
  workflow_dispatch: {}

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: checkout
        uses: actions/checkout@master
      - name: use Node 18
        uses: actions/setup-node@v1
        with:
          node-version: 18
      - name: npm install
        run: |
          npm install -g hexo-cli
          npm install
        env:
         CI: true
      - name: hexo build
        run: |
          hexo clean
          hexo generate
        env:
          CI: true
      - name: Deploy
        uses: easingthemes/ssh-deploy@v2.0.7
        env:
          SSH_PRIVATE_KEY: ${{ secrets.ACCESS_TOKEN }}
          ARGS: "-avz --delete"
          SOURCE: "public/"
          REMOTE_HOST: ${{ secrets.REMOTE_HOST }}
          REMOTE_USER: ${{ secrets.REMOTE_USER }}
          TARGET: ${{ secrets.TARGET }}
```

文件中#开头的是注释，不用管。

开始的 `name` 表示名字，给这个workflow起一个名字， `on` 表示的是在什么阶段触发这个工作流， `push` 就代表在提交代码的时候，并且分支是 `main` ， `workflow_dispatch` 表示的是可以在Github仓库的界面手动执行这个workflow， `{}` 表示的是空参数，一般手动执行可以带一些参数。 `jobs` 表示执行的任务，一个workflow可以有多个job， `runs-on` 表示运行环境， `steps` 是任务中具体的步骤，里面的每一个 `-` 代表了一个 `action` ，其中 `action` 也可以有自己的 `name` ，也可以使用 `uses` 使用别人写好的 `action` 。那怎么看有哪些 `actions` 呢？可以看这里，使用别人的 `action` 的格式是 `uses: 用户名/action名称@版本号` 。

上面 的部分就相当于Github在你提交的时候，用一个机器（ubuntu-latest）运行你给的指令。

最重要的就是最后一部分了，下面精讲：

我们发现这里有四个变量 `ACCESS_TOKEN` 、 `REMOTE_HOST` 、 `REMOTE_USER` 、 `TARGET` ，分别代表的是服务器Git的私钥，服务器的IP，服务器的用户名，最后打包的服务器的位置，举个栗子： `147.0.130.190` 、 `root` 、 `/www/root/blog` ，最关键的是第一个变量，

那么这个值是什么呢？首先去你服务器的 `~/.ssh` 目录，此时目录下应该有4个文件，分别是 `authorized_keys` 、 `id_rsa` 、 `id_rsa.pub` 、 `known_hosts` 。如果没有 `id_rsa` 和 `id_rsa.pub` 的，可以使用 `ssh-keygen` 来生成，这两个文件就是安装Git时需要生成的私钥和公钥。这个时候你看看 `authorized_keys` 里面有没有内容，如果有内容说明你之前设置过， `ACCESS_TOKEN` 的值就是 `authorized_keys` 所对应的私钥。如果没有内容的话，你可以直接设置为公钥 `id_rsa.pub` 的内容，如执行命令 `cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys` ，此时就会把 `id_rsa.pub` 的内容写入 `authorized_keys` 中，然后把 `ACCESS_TOKEN` 的值设置为私钥 `id_rsa` 中的内容，你可以运行命令 `cat ~/.ssh/id_rsa` 然后把内容复制一份到 `ACCESS_TOKEN` 中，如下：

最后在哪里设置?

注意不要设置到environment secrets了，我之前就设置错了，到这里，就大功告成了！！！

如果你遇到了类似这样的问题：

```shell
⚠ [Rsync] error: rsync exited with code 255

⚠ [Rsync] stderr:  Warning: Permanently added '***' (RSA) to the list of known hosts.

```

那说明你的服务器与Github服务器不能ssh通信，或者没有相应的权限，记得开放你的22端口，如果你不放心22端口，可以给Github的IP设置白名单，至此我的Github Action就设置好了。看你的啦！

标签： GitHub Hexo 话题： 每天一个知识点

评论 1

登录 / 注册 即可发布评论!

最热 | 最新

理子 躺平爱好者

写的挺好 actions插件版本可以再更新一下

7月前 点赞 评论

为你推荐

Github actions 重塑你的 Blog Workflows

Jayconscious | 3年前 | 593 | 3 | 评论 Vue.js GitHub

04.使用 github actions+docker 自动部署前后端分离项目 zhontai (.net core+vue)

易墨 | 1年前 | 337 | 点赞 | 评论 后端 GitHub Docker

使用 Github Actions 实现一个简单的 ci/cd

五块木头 | 3年前 | 4.2k | 18 | 评论 前端 GitHub

如何优雅实现自动化部署

舟鱼 | 2年前 | 1.4k | 13 | 2 GitHub VuePress

如何写一个其他人可以使用的GitHub Action

晚风予星 | 1年前 | 779 | 4 | 评论 前端 GitHub 开源

Github Actions 自动化部署 Vue3 博客项目

小猪皮皮呆 | 3年前 | 2.5k | 23 | 4 CI/CD

GitHub Actions实现定时任务，免费运行

你不会困 | 1年前 | 767 | 13 | 8 Node.js

利用 GitHub Action 对项目进行自动部署

楷鹏Dev | 2年前 | 4.2k | 2 | 评论 | GitHub

带你学习通过GitHub Actions如何快速构建和部署你自己的项目，打造一条属于自己的流水线

初夏的阳光 | 1年前 | 12k | 36 | 25 | GitHub Docker Java

探秘 GitHub Actions 自动化部署全流程

前端日常开发 | 3月前 | 96 | 1 | 评论 | 前端

打造Github Issue到Hexo部署自动工作流🔥🔥

Geek技术前线 | 3年前 | 1.4k | 5 | 1 | 前端 GitHub

进来拷贝，来用GitHub Actions部署前端项目

whoopshzh | 3年前 | 1.1k | 9 | 6 | 前端 GitHub

基于Github Actions 和 Portainer 实现CI/CD

内秘心书 | 5月前 | 152 | 2 | 评论 | CI/CD

GitHub自动化部署(CD) asp.net core 5.0 项目到Heroku平台

华林 | 3年前 | 348 | 点赞 | 评论 | GitHub 后端

博客搭建 | 四、GitHub Actions ssh-deploy部署至阿里云

夜茶 | 1年前 | 572 | 1 | 评论 | 前端 GitHub