链滴

GitHub Action

作者：someone27889

原文链接：https://ld246.com/article/1588746808443

来源网站：链滴

WorkFlow

GithubWorkFlow就是git自带的CI服务

WorkFlow文件

```
name: Greet Everyone
# 事件
on: [push]
# 工作集合
jobs:
  build:
    # 作业名称为 Greeting
    name: Greeting
    # 此作业在 Linux 上运行,runs-on选项一共就有几种类型
    runs-on: ubuntu-latest
    steps:
      # 此步骤使用 GitHub 的 hello-world-javascript-action：https://github.com/actions/hello-world-javascript-action
      - name: Hello world
        uses: actions/hello-world-javascript-action@v1
        with:
          who-to-greet: 'Mona the Octocat'
        id: hello
      # 此步骤打印上一步操作的输出（时间）。
      - name: Echo the greeting's time
        run: echo 'The time was ${{ steps.hello.outputs.time }}.'
```

工作流程语法

原文链接：GitHub Action

工作流程语法原文

- name

工作流程的名称

- on

触发时机,push,pull_request,create更多

具有活动类型或配置的多个事件示例

```
on:
  push:
    branches:
     # master分支发push请求时
      - master
  pull_request:
    branches:
     # master分支pull_request时
      - master
  page_build:
  release:
    types:
      - created
```

- on.<event_name>.types

```
# 当release的published,created,edited事件被触发时
on:
  release:
    types: [published,created,edited]
```

- on.<push|pull_request>.<brancheds|tags>

```
# 在这些分支和tag被push代码时触发
# ignore在这些分支和tag被push时不会触发
on:
  push:
    branches:
      - master
      - 'mona/octocat'
      - 'releases/**'
    tags:
      - v1
      - v1.*
    branches-ignore:
      - devp
      - '**-alpha'
    tags-ignore:
      - v2.*
```

- on.<push|pull_request>.paths

```
# 当有js被推送时触发
# 当docs下推送时不触发
```

原文链接：GitHub Action

```
on:
  push:
    paths:
    - '**.js'
    paths-ignore:
    - 'docs/**'
```

- on.schedule

POSIX cron

```
# 每隔15分钟构建一次
on:
  schedule:
     - cron: '*/15 * * * *'
```

- env

```
env:
  SERVER: production
```

- jobs

工作流程运行包括一项或多项作业。 作业默认是并行运行。 要按顺序运行作业，您可以使用

- jobs.<job_id>(.name)

```
jobs:
   my_first_job:
      name: MyfirstJob
   my_second_job:
      name: MysecondJob
```

- jobs.<job_id>.needs

job的依赖关系

```
jobs:
   job1:
   job2:
      needs: job1
   job3:
      needs: [job1,job2]
```

- jobs.<job_id>.runs-on

<table><thead><tr><th>虚拟环境</th><th>YAML 工作流程标签</th></tr></thead><tbody><tr><td>Windows Server 2019</td><td><code>windows-latest</code> 或 <code>windows-2019</code></td></tr><tr><td>Ubuntu 18.04</td><td><code>ubuntu-latest</code> 或 <code>ubuntu-18.04</code></td></tr><tr><td>Ubuntu 16.04</td><td><code>ubuntu-16.04</code></td></tr><tr><td>macOS Catalina 10.15</td><td><code>macos-latest</code> or <code>macos-10.15</code></td></tr></tbody></table>

```
runs-on : windows-latest
```

原文链接：GitHub Action

● jobs.<job_id>.env

```
jobs:
  job1:
    env:
      FIRST_NAME: mona
```

● jobs.<job_id>.if

● jobs.<job_id>.steps

为作业定义的变量在作业执行时将覆盖名称相同的工作流程变量。步骤可以运行命令、运行设置任务或者运行您的仓库、公共仓库中的操作或 Docker 注册表中发布的操作。 并非所有步骤都会运行操作但所有操作都会作为步骤运行。

```
name: Greeting from Mona

on: push

jobs:
  my-job:
    name: My Job
    runs-on: ubuntu-latest
    steps:
    - name: Print a greeting
      env:
        MY_VAR: Hi there! My name is
        FIRST_NAME: Mona
        MIDDLE_NAME: The
        LAST_NAME: Octocat
      run: |
        echo $MY_VAR $FIRST_NAME $MIDDLE_NAME $LAST_NAME.
```

● jobs.<job_id>.stemp2.uses

可以使用工作流程所在仓库中、公共仓库中或发布Docker容器镜像中定义的操作

```
jobs:
  my_first_job:
    steps:
      - name: My first step
        uses: actions/heroku@master
      - name: My second step
        uses: actions/aws@v2.0.1


jobs:
  my_first_job:
    steps:
      - name: My first step
        uses: docker://alpine:3.8
```

● jobs.<job_id>.steps.run

使用shell,working-directory:指定命令的工作目录,shell指定cmd环境

```
- name: install dependencies and build
  run: |
    npm ci
  shell: bash



- name: Clean temp directory
  run: rm -rf *
  working-directory: ./temp
```

环境列表`<table><thead><tr><th>支持的平台/th><th><code>shell</code> 参数</th><th>描述</th><th>内运行命令</th></tr></thead><tbody><tr><td>所有</td><td><code>bash</code></td><td>Windows 平台上回退到 <code>sh</code> 的默认 shell。 指定 Windows 上的 bash shell 时，使用 Git for Windows 随附的 bash shel。 </td><td><code>bash --noprofile -norc -eo pipefail {0}</code></td></tr><tr><td>所有</td><td><code>pwsh</code></td><td>PowerShell Core。 GitHub 将扩展名 <code>.ps1</code> 附加到您的脚本名称。 </td><td><code>pwsh -command "& '{0}'"</code></td></tr><tr><td>所有</td><td><code>python</code></td><td>运行 python 命令。 </td><td><code>python {0}</code></td></tr><tr><td>Linux / macOS</td><td><code>sh</code></td><td>提供 shell 且 在路径中找不到 <code>bash</code> 时的非 Windows 平台的后退行为。 </td><td><code>sh -e {0}</code></td></tr><tr><td>Windows</td><td><code>cmd</code></td><td>GitHub 将扩展名 <code>.cmd</code> 附加到您的脚本名称并替换 <code>{0}</code>。 </td><td><code>%ComSpec% /D /E:ON /V:OFF /S /C "CALL "{0}""</code>.</td></tr><tr><td>Windows</td><td><code>powershell</code></td><td>是 Windows 上使用的默认 shell。 Desktop PowerShell。 GitHub 将扩展名 <code>.ps1</code> 附加到您的脚本名称。 </td><td><code>powershell -command "& '{0}'"</code>.</td></tr></tbody></table>

● jobs.<job_id>.steps.with

传入参数

```
# with传入了actions/hello_word@master操作所需要的三个参数,参数名称为fitst_name,middle_name,last_name
jobs:
  my_first_job:
    steps:
      - name: My first step
        uses: actions/hello_world@master
        with:
          first_name: Mona
          middle_name: The
          last_name: Octocat
```

● jobs.<job_id>.steps.with.args

docker容器参数

```yaml
steps:
  - name: Explain why this job ran
    uses: monacorp/action-name@master
    with:
      entrypoint: /bin/echo
      args: The ${{ github.event_name }} event triggered this step.
```

● jobs.<job_id>.steps.with.entrypoint

覆盖dockerfile中的 docker entrypoint

```yaml
steps:
  - name: Run a custom command
    uses: monacorp/action-name@master
    with:
      entrypoint: /a/different/executable
```

● jobs.<job_id>.steps.env

设置整个工作流程或某个作业的环境变量。

公共操作可在自述文件中指定预期的环境变量。 如果要在环境变量中设置密码，必须使用 secrets上文设置

```yaml
steps:
  - name: My first action
    env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      FIRST_NAME: Mona
      LAST_NAME: Octocat
```

● jobs.<job_id>.steps.continue-on-error

错误跳过

```yaml
steps:
  - continue-on-error: true
  - name: My first action
    env:
      GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
      FIRST_NAME: Mona
      LAST_NAME: Octocat
```

● jobs.<job_id>.steps.timeout-minutes

最大运行时间

● jobs.<job_id>.strategy

矩阵，比如 3*3实际是9个任务

```yaml
# 3*1 3个任务，每个都会试
strategy:
  matrix:
    node: [6, 8, 10]
steps:
```

```
  # Configures the node version used on GitHub-hosted runners
  - uses: actions/setup-node@v1
    with:
      # The Node.js version to configure
      node-version: ${{ matrix.node }}


# os和node全排列
runs-on: ${{ matrix.os }}
strategy:
  matrix:
    os: [ubuntu-16.04, ubuntu-18.04]
    node: [6, 8, 10]
steps:
  - uses: actions/setup-node@v1
    with:
      node-version: ${{ matrix.node }}
```

添加额外的配置到矩阵中

```
# include 在os为windows-latest时且node为4时，npm为2
runs-on: ${{ matrix.os }}
strategy:
  matrix:
    os: [macos-latest, windows-latest, ubuntu-18.04]
    node: [4, 6, 8, 10]
    include:
      - os: windows-latest
        node: 4
        npm: 2
```

● jobs.<job_id>.strategy.fail-fast

任何matrix作业失败,取消所有进行中的作业,默认为true

● jobs.<job_id>.strategy.max-parallel

使用matrix作业策略时同时运行的最大作业数,默认情况下,github将最大并发运行的作业数量

```
strategy:
  max-parallel: 2
```

● jobs.<job_id>.container

用于运行作业中尚未指定容器的任何步骤的容器。

```
jobs:
  my_job:
    container:
      image: node:10.16-jessie
      env:
        NODE_ENV: development
      ports:
        - 80
      volumes:
```

```
      - my_docker_volume:/volume_mount
    options: --cpus 1
```

● jobs.<job_id>.container.options

附加docker容器资源选项

● jobs.<job_id>.services

为ci提供服务

```
services:
  nginx:
    image: nginx
    volumes:
      - my_docker_volume:/volume_mount
      - /data/my_data
      - /source/directory:/destination/directory
    ports:
      - 8080:80
  redis:
    image: redis
    ports:
      - 6379/tcp
```

非卖品

项目地址：https://github.com/ferried/animal-crossing-cli

action地址： https://github.com/ferried/animal-crossing-cli/runs/648812568?check_suite_focs=true

```
name: Go
on:
  push:
    branches: [ devp ]
  pull_request:
    branches: [ devp ]
jobs:
  release:
    name: release
    runs-on: ubuntu-latest
    steps:

      - name: setup go
        uses: actions/setup-go@v2
        with:
          go-version: ^1.13
          id: go

      - name: checkout code
        uses: actions/checkout@v2
```

```
- name: install dependencies
  run: |
    go get -v -t -d ./...
    if [ -f Gopkg.toml ]; then
        curl https://raw.githubusercontent.com/golang/dep/master/install.sh | sh
        dep ensure
    fi

- name: test code
  run: go test
```