

# Github Actions

吴温炎

GO →



精彩继续!

# 这里有更多前沿技术与工程实现

加入我们，你将可以收获：



分享人



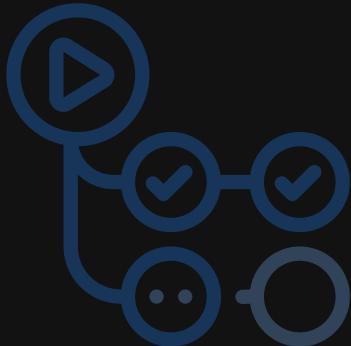
参与人



# 说 明

本 内 容 适 合 新 手 或 初 入 开 源 领 域 的 人

# TOC



1. 基本介绍
2. 上手指南
3. 应用场景
4. 高级特性
5. 最佳实践
6. 案例分享
7. QA

# 基本介绍

- Github Actions 是什么?
- Features
- Pricing
- 简单演示

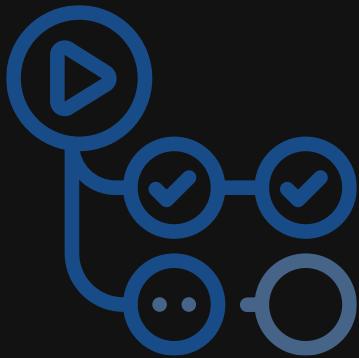


# 基本介绍 Github Actions 是什么？

先看个东西：

**Features** [Actions](#) [Packages](#) [Security](#) [Codespaces](#) [Copilot](#) [Code review](#) [Search](#) [Issues](#) [Discussions](#)

来源：<https://github.com/features/actions>



## Automate your workflow from idea to production

从想法到生产，自动化您的工作流程

GitHub Actions makes it easy to automate all your software workflows, now with world-class CI/CD. Build, test, and deploy your code right from GitHub. Make code reviews, branch management, and issue triaging work the way you want.

GitHub Actions 现在可以使用世界一流的 CI/CD 轻松实现所有软件工作流程的自动化。直接从 GitHub 构建、测试和部署您的代码。按照您想要的方式进行代码审查、分支管理和问题分类。

- 首个公测版，发布于 [2018 年 10 月](#)
- 在此之前 Github 的 CI/CD 以第三方服务 [Travis CI](#) 居多。

# 基本介绍 Features

- **操作系统** - 适用于 Linux、macOS、Windows、ARM 和容器；直接在虚拟机或容器内运行。
- **矩阵构建** - 通过矩阵工作流程可以同时跨多个操作系统和运行时版本进行测试，从而节省时间。
- **任何语言** - 支持 Node.js、Python、Java、Ruby、PHP、Go、Rust、.NET 等。
- **实时日志** - 查看您的工作流程实时运行的颜色和表情符号。
- **秘钥存储** - 通过将包含 Git 流程的工作流程文件编码到您的存储库中，自动执行您的软件开发实践。
- **多容器测试** - compose 只需将一些内容添加到工作流程文件中即可在工作流程中测试您的 Web 服务及其数据库。
- **社区支持的工作流程** - GitHub Actions 连接您的所有工具，以自动化开发工作流程的每一步。

# 基本介绍 Pricing

- 免费版 - 2000 分钟/月
- Pro - 3000 分钟/月 (其中 Pro 为 4 美元/月)
- Team - 3000 分钟/月 (其中 Team 为 4 美元/用户/月)
- Enterprise - 50000 分钟/月 (其中 Enterprise 为 21 美元/用户/月)

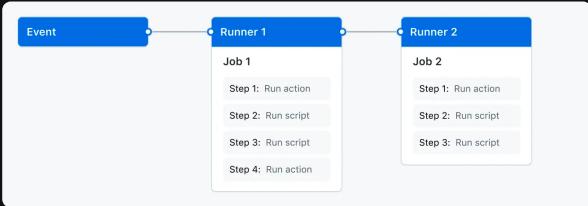
对于 **开源** 项目，无限制。

开源就可以薅羊毛.....

---

详见：[个人计划](#) | [组织计划](#)

# 基本介绍 基础概念



```
on: push
jobs:
  test:
    strategy:
      matrix:
        platform: [ubuntu-latest, macos-latest, windows-latest]
    runs-on: ${{ matrix.platform }}
    steps:
    - uses: actions/checkout@v3
    - uses: actions/setup-node@v3
      with:
        node-version: 16
    - run: npm run test
```

工作流 **workflows** 自动化流程，可以有多个，存放在 `.github/workflows` 目录中。

事件 **event** 比如推送代码，创建 PR 等；支持的事件

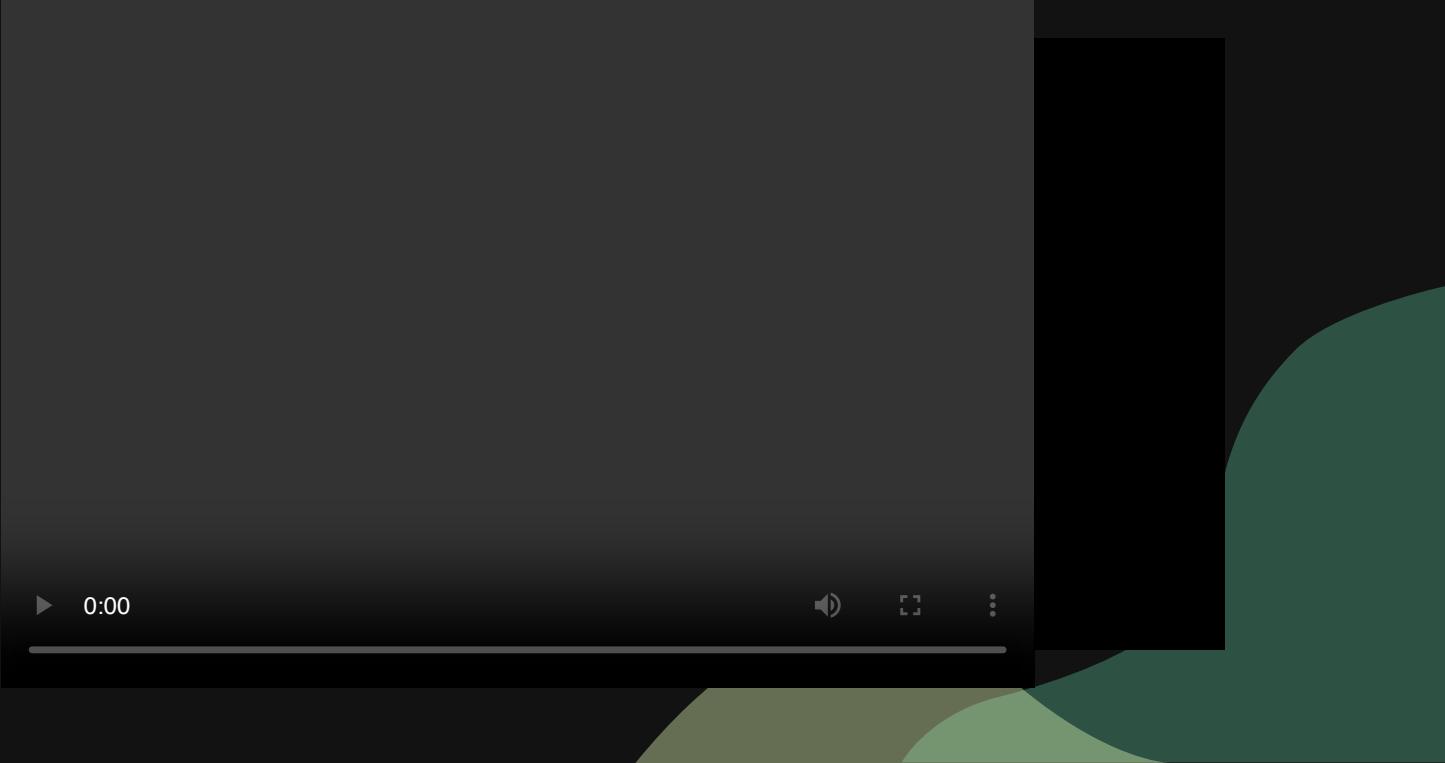
作业 **jobs** 在同一运行器上执行的一组步骤。顺序执行，相互依赖。

操作 **actions** 作业中的一组任务，可以是自定义的，也可以是开源市场提供的。

运行程序 **runners** 运行工作流的服务器；支持 Ubuntu Linux、Microsoft Windows 和 macOS。

# 基本介绍

## 简单演示



# 上手指南

- 创建工作流( `workflows` )
- `workflows` 文件的结构和语法
- 触发器和事件
- 任务和步骤
- 使用环境变量和密钥



# 上手指南 创建 workflows

- 方式一：使用官方模板（适合新手）

The screenshot shows the GitHub Actions interface. On the left, there's a sidebar with 'Actions' selected, and a 'New workflow' button highlighted with a red circle. The main area shows a template named 'blank.yml' with the following YAML code:

```
1 # This is a basic workflow to help you get started with Actions
2
3 name: CI
4
5 # Controls when the workflow will run
6 on:
7   # Triggers the workflow on push or pull request events but only for the "main" branch
8   push:
9     branches: [ "main" ]
10    pull_request:
11      branches: [ "main" ]
12
13 # Allows you to run this workflow manually from the Actions tab
14 workflow_dispatch:
15
16 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
17 jobs:
18   # This workflow contains a single job called "build"
19   build:
20     # The type of runner that the job will run on
21     runs-on: ubuntu-latest
22
23 # Steps represent a sequence of tasks that will be executed as part of the job
24 steps:
25   # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
26   - uses: actions/checkout@v3
27
28   # Runs a single command using the runners shell
29   - name: Run a one-line script
30     run: echo Hello, world!
31
32   # Runs a set of commands using the runners shell
33   - name: Run a multi-line script
34     run:
35       echo Add other actions to build,
36       echo test, and deploy your project.
37
```

- 方式二：手动创建（适合有经验的用户）

文件路径： `.github/workflows/[xxx].yml`，支持配置多个 `workflows` 文件。

```
name: CI

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Run a one-line script
        run: echo Hello, world!

      - name: Run a multi-line script
        run: |
          echo Add other actions to build,
          echo test, and deploy your project.
```

# 上手指南 workflows 文件的结构和语法

```
# 工作流的名称
name: CI

# 触发器
on:
  # 当往 main 分支推送代码, 或发起合并请求时触发
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

# 任务列表 (此处只列了一个)
jobs:
  # 任务名称, 比如此处为 build 任务; 多个任务默认是并行, 可通过配置建立依赖关系
  build:
    # 任务运行的服务器环境
    runs-on: ubuntu-latest

# 任务运行的步骤, 此处是串行
steps:
  # 拉取代码, 此处使用的时第三方 (虽然是官方组织) 的 actions 组件
  - uses: actions/checkout@v3

  # 执行一个单行命令 (其中 name 为该命令的名称)
```

Code Issues Pull requests Actions Projects Settings

← CI 工作流名

Create blank.yml #1

Summary

Triggered via push 4 hours ago  
f1c1125 pushed → 5b8f613 main

Status Success Total duration 13s Billable time 1m Artifacts -

Jobs

build

Run details

Usage

Workflow file

blank.yml  
on: push

build 4s

The screenshot shows the GitHub Actions interface for a repository. At the top, there are navigation links: Code, Issues, Pull requests, Actions (which is highlighted with an orange underline), Projects, and Settings. Below this, a red arrow points from the 'CI' link to the '工作流名' (Workflow Name) label. Another red arrow points from the 'build' job entry in the 'Jobs' section to the 'build' step in the expanded workflow details. The workflow name is 'blank.yml' and it triggers on 'push'. The single job, also named 'build', has a status of 'Success' and a duration of '13s'. The 'Workflow file' section shows the YAML configuration for the workflow.

## build

succeeded 4 hours ago in 4s

>  Set up job

✓  Run actions/checkout@v3

```
1 ► Run actions/checkout@v3
14 Syncing repository: github-actions-templates/blank
15 ► Getting Git version info
19 Temporarily overriding HOME='/home/runner/work/_temp/4518519d-4856-4c1d-a53d-9330f8c03d7e' before making global git config changes
20 Adding repository directory to the temporary git global config as a safe directory
21 /usr/bin/git config --global --add safe.directory /home/runner/work/blank/blank
22 Deleting the contents of '/home/runner/work/blank/blank'
23 ► Initializing the repository
37 ► Disabling automatic garbage collection
39 ► Setting up auth
45 ► Fetching the repository
62 ► Determining the checkout info
63 ► Checking out the ref
67 /usr/bin/git log -1 --format=%H
68 '5b8ff6138f7722cc8c0a4208bf0751a4c72bd55ae'
```

✓  Run a one-line script

```
1 ► Run echo Hello, world!
4 Hello, world!
```

✓  Run a multi-line script

```
1 ► Run echo Add other actions to build,
5 Add other actions to build,
6 test, and deploy your project.
```

>  Post Run actions/checkout@v3

>  Complete job

# 上手指南 触发器和事件

工作流程触发器是导致工作流程运行的事件。这些事件可以是：

- 工作流存储库中发生的事件，如：推送代码、创建 ISSUE、发起 PR
- 在 GitHub 之外发生并在 GitHub 上触发 `repository_dispatch` 事件的事件，如：通过 Webhook 发送自定义时间
- 预定时间，如：每天 4 点全量单测
- 手动，如：点击按钮

例如：当你推送代码到某个分支时，可以触发事件，以运行相关任务。

```
on:  
  issues:  
    types:  
      - opened  
  
jobs:  
  label_issue:  
    runs-on: ubuntu-latest  
    steps:  
      - env:  
        GITHUB_TOKEN: ${{ secrets.MY_TOKEN }}  
        ISSUE_URL: ${{ github.event.issue.html_url }}  
      run: /  
        gh issue edit $ISSUE_URL --add-label "triage"
```

配合官方的 `gh` 命令工具效果更佳。

## 常用工作流事件

- 推送代码，创建合并等

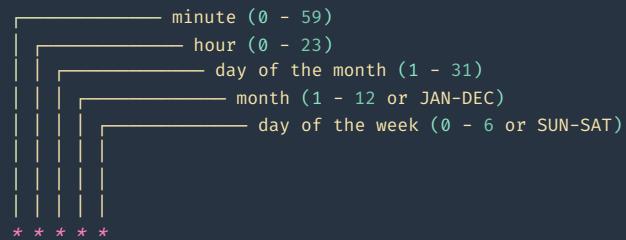
```
on:  
  push:  
    branches:  
      - 'main'  
      - 'releases/**' # 指定分支  
    tags:  
      - 'v1.*' # 指定 Tag  
    paths:  
      - '**.js' # 指定文件  
      - '!**-alpha.js' # 排除文件  
  pull_request:  
    branches:  
      - 'main'  
      - '!releases/**-alpha' # 忽略分支
```

## 常用工作流事件

- 定时事件

```
on:  
schedule:  
- cron: '30 5 * * 1,3'  
- cron: '30 5 * * 2,4'
```

Cron 语法:



## 工作流事件

- 更多事件: <https://docs.github.com/zh/actions/using-workflows/events-that-trigger-workflows>
- 基础概念: <https://docs.github.com/zh/actions/using-workflows/triggering-a-workflow>
- 手动运行工作流程: <https://docs.github.com/zh/actions/using-workflows/manually-running-a-workflow>
- 工作流语法: <https://docs.github.com/zh/actions/using-workflows/workflow-syntax-for-github-actions>

# 上手指南 任务和步骤

- 工作流运行由一个或多个 `jobs` 组成，默认情况下 **并行** 运行。
- 可使用 `jobs.<job_id>.needs` 关键字定义对其他作业的依赖关系，来实现顺序运行。
- 每个作业在 `runs-on` 指定的运行器环境中运行。

一个 🥞

案例地址

```
name: Test Jobs

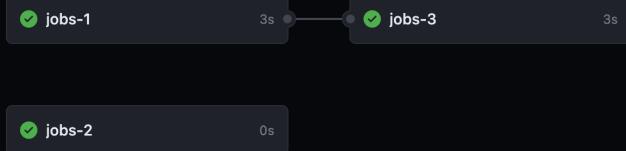
on:
  workflow_dispatch:
    inputs:
      test-jobs:
        description: '测试 Jobs'

jobs:
  jobs-1:
    runs-on: ubuntu-latest
    steps:
      - name: Run a one-line script
        run: echo Hello, world! This is jobs-1

  jobs-2:
    runs-on: ubuntu-latest
    steps:
      - name: Run a one-line script
        run: echo Hello, world! This is jobs-2

  jobs-3:
    runs-on: ubuntu-latest
```

jobs.yml  
on: workflow\_dispatch



# 上手指南 使用环境变量和密钥

[案例地址](#)

```
name: Test Env

on:
  workflow_dispatch:
    inputs:
      test-jobs:
        description: '测试 Env'

env:
  ENVIRONMENT: prod

jobs:
  dev:
    runs-on: ubuntu-latest
    env:
      ENVIRONMENT: dev
    steps:
      - run: echo The env is ${{ env.ENVIRONMENT }}

  prod:
    runs-on: ubuntu-latest
    steps:
      - run: echo The env is ${{ env.ENVIRONMENT }}
```

The screenshot shows the GitHub Actions interface with two separate workflows: 'dev' and 'prod'.  
**dev Workflow:**

- Jobs:** dev (Succeeded)
- Run details:** Set up job, Run echo The env is dev (Step 1: Run echo The env is dev, Step 6: The env is dev)
- prod Workflow:** (Succeeded)
  - Jobs:** prod (Succeeded)
  - Run details:** Set up job, Run echo The env is prod (Step 1: Run echo The env is prod, Step 6: The env is prod)

## 补充

- 环境变量你可以理解为执行了: `export ENVIRONMENT=prod`
- 支持编程语言中的获取, 如: Go 语言的 `os.LookupEnv`

The screenshot shows a CI pipeline interface with a dark theme. On the left, there is a code editor window displaying a Go program. On the right, there is a log viewer window showing the execution steps of a job named "code".

**code**  
succeeded 3 minutes ago in 22s

Execution steps:

- >  Set up job
- >  Checkout code
- >  Install Go
- >  Run Go
- >  Post Install Go
- >  Post Checkout code
- >  Complete job

Log output for the "Run Go" step:

```
1 ► Run cd env && go run main.go
6 prod
```

# 如果我要输入我的银行卡密码怎么办？

- 路径: Settings → Security → Actions secrets and variables → Actions → New repository secret

The image shows a composite view of the GitHub Settings interface and a modal window for creating a new repository secret.

**Left Side (GitHub Settings):**

- Header:** Settings (marked with a red circle containing the number 1)
- Left sidebar:** General, Access, Collaborators and teams, Code and automation, Branches, Tags, Rules, Actions (marked with a red circle containing the number 2), Webhooks, Pages, Security, Code security and analysis, Deploy keys, Secrets and variables (marked with a red circle containing the number 3), Actions, Codespaces, Dependabot, Integrations, GitHub Apps, Email notifications.
- Central Content:** **Actions secrets and variables**
  - Secrets and variables allow you to manage reusable configuration data. Secrets are encrypted and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for non-sensitive data. [Learn more about variables](#).
  - Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.
- Modal (Top Right):** **Actions secrets / New secret**
  - Name \***: BANK\_PASSWORD
  - Secret \***: 123456
  - Add secret** button

```
name: Test Env

on:
  workflow_dispatch:
    inputs:
      test-jobs:
        description: '测试 Env'

env:
  PWD: ${{ secrets.BANK_PASSWORD }}

jobs:
  pwd:
    runs-on: ubuntu-latest
    steps:
      - run: echo The bank password is ${{ env.PWD }}
```

pwd

succeeded 2 minutes ago in 0s

✓ Set up job

- 1 Current runner version: '2.309.0'
- 2 ► Operating System
- 6 ► Runner Image
- 11 ► Runner Image Provisioner
- 13 ► GITHUB\_TOKEN Permissions
- 17 Secret source: Actions
- 18 Prepare workflow directory
- 19 Prepare all required actions
- 20 Complete job name: pwd

✓ Run echo The bank password is \*\*\*

- 1 ► Run echo The bank password is \*\*\*
- 7 The bank password is \*\*\*

> ✓ Complete job

## 适用场景：

- 单元测试：API 秘钥、地址等
- 第三方服务：单测覆盖率（如：`codecov`）
- 敏感信息：密码等
- .....

# 应用场景

- 自动化构建和测试
- 自动化部署
- 定时任务和计划作业
- 集成第三方服务



# 应用场景

## 自动化构建和测试

案例地址

```
// package
package example

func Example() string {
    return "Hello, world!"
}

// package test
package example

import "testing"
```

```
name: Go Test
on:
  push:
    branches: [ main ]
    paths:
      - "go/**"
env:
  GOPROXY: "https://proxy.golang.org"

jobs:
  test:
```

go test

succeeded 4 minutes ago in 24s

Search logs

Set up job 4s

Checkout code 1s

Install Go 0s

Run actions/setup-go@v3

Setup go version spec

go version go1.20.8 linux/amd64

go env

Run go version 0s

Run go version

go version go1.20.8 linux/amd64

Run tests 15s

cd go && go test ./... -v -covermode=atomic -race -coverprofile=coverage.txt

==== RUN TestExample

--- PASS: TestExample (0.00s)

PASS

example coverage: 100.0% of statements

example\_0.024s coverage: 100.0% of statements

# 应用场景

## 自动化部署

案例地址

```
name: Github Pages

on:
  push:
    branches:
      - main

jobs:
  pages:
    runs-on: ubuntu-latest
    permissions:
      contents: write
    steps:
      - uses: actions/checkout@v3

      - uses: actions/setup-node@v3
        with:
          node-version: 18
          cache: 'npm'

      - uses: actions/cache@v3
        with:
          path: node_modules
          key: ${{ runner.OS }}-npm-cache-${{ hashFiles(
```

pages

succeeded 3 weeks ago in 15s

Search logs

Set up job

Run actions/checkout@v3

Run actions/checkout@v3

Run actions/setup-node@v3

```
1 ► Run actions/checkout@v3
14 ► Syncing repository: github-actions-templates/hexo
15 ► Getting Git version info
19 Temporarily overriding HOME='/home/runner/work/_temp/53dceec7-a914-4d8a-ac5d-e4bd76952a5a' before making global git config changes
20 Adding repository directory to the temporary git global config as a safe directory
21 /usr/bin/git config --global --add safe.directory /home/runner/work/hexo/hexo
22 Deleting the contents of '/home/runner/work/hexo/hexo'
23 ► Initializing the repository
37 ► Disabling automatic garbage collection
39 ► Setting up auth
45 ► Fetching the repository
89 ► Determining the checkout info
--> Run actions/checkout@v3
--> Run actions/setup-node@v3
1 ► Run actions/setup-node@v3
8 Found in cache @ /opt/hostedtoolcache/node/18.17.1/x64
9 ► Environment details
```

## 效果地址

The screenshot shows a blog homepage with a dark background featuring a starry night sky at the top. The title "Hexo" is centered in a large, white, sans-serif font. Below the title, there's a date "2023-08-29". The main content area contains a post titled "Hello World" with a brief introduction. To the right, there are two sidebar boxes: "ARCHIVES" showing "August 2023" and "RECENT POSTS" showing "Hello World". At the bottom left, there's a terminal-style code block with the command "\$ hexo new "My New Post"".

Home Archives

# Hexo

2023-08-29

## Hello World

Welcome to [Hexo](#)! This is your very first post. Check [documentation](#) for more info. If you get any problems when using Hexo, you can find the answer in [troubleshooting](#) or you can ask me on [GitHub](#).

### Quick Start

### Create a new post

```
1 $ hexo new "My New Post"
```

**ARCHIVES**

August 2023

**RECENT POSTS**

Hello World

# 应用场景 定时任务和计划作业

案例地址

```
name: Cron
on:
  schedule:
    - cron: "0 0 * * *"
    - cron: "0 12 * * *"
    - cron: "*/5 * * * *"
  env:
    GOPROXY: "https://proxy.golang.org"
jobs:
  test:
    name: "go test"
    runs-on: ubuntu-latest
  steps:
    - name: Checkout code
      uses: actions/checkout@v3
```

Cron	Event	Status	Branch	Actor
Cron	1 minute ago	⌚ 32s	...	
Cron	12 minutes ago	⌚ 32s	...	
Cron	23 minutes ago	⌚ 28s	...	
Cron	34 minutes ago	⌚ 42s	...	

实际操作下来，并不一定非常准时。（可能是 Queue 机制）

说明：

- 时区相关：基于 UTC 时区
- 最短支持：每 5 分钟一次

# 应用场景 集成第三方服务

```
name: Go Test
on:
  push:
    branches: [ main ]
    paths:
      - "go/**"
env:
  GOPROXY: "https://proxy.golang.org"
jobs:
  test:
    name: "go test"
    runs-on: ubuntu-latest
  steps:
    - name: Checkout code
      uses: actions/checkout@v3
    - name: Install Go
      uses: actions/setup-go@v3
    - run: go version
    - name: Run tests
      run: go test ./... -v -covermode=atomic -race -c
```

Marketplace / Search results

Types

Actions

Categories

API management

Chat

Code quality

Code review

Continuous integration

Dependency management

Deployment

IDEs

Learning

Search for apps and actions

Sort: Best Match

Actions

An entirely new way to automate your development workflow.

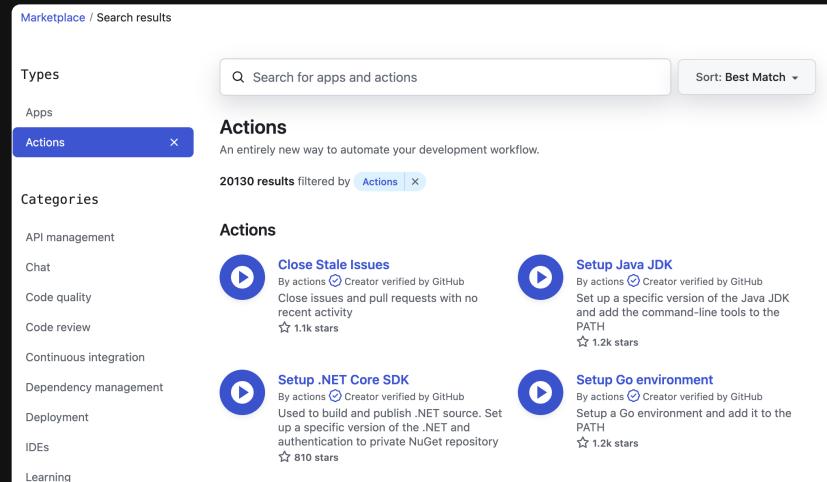
20130 results filtered by Actions

**Close Stale Issues**  
By actions Creator verified by GitHub  
Close issues and pull requests with no recent activity  
1.1k stars

**Setup Java JDK**  
By actions Creator verified by GitHub  
Set up a specific version of the Java JDK and add the command-line tools to the PATH  
1.2k stars

**Setup .NET Core SDK**  
By actions Creator verified by GitHub  
Used to build and publish .NET source. Set up a specific version of the .NET and authentication to private NuGet repository  
810 stars

**Setup Go environment**  
By actions Creator verified by GitHub  
Setup a Go environment and add it to the PATH  
1.2k stars



## GitHub Actions 第三方服务:

- Marketplace: <https://github.com/marketplace?type=actions>

# 高级特性

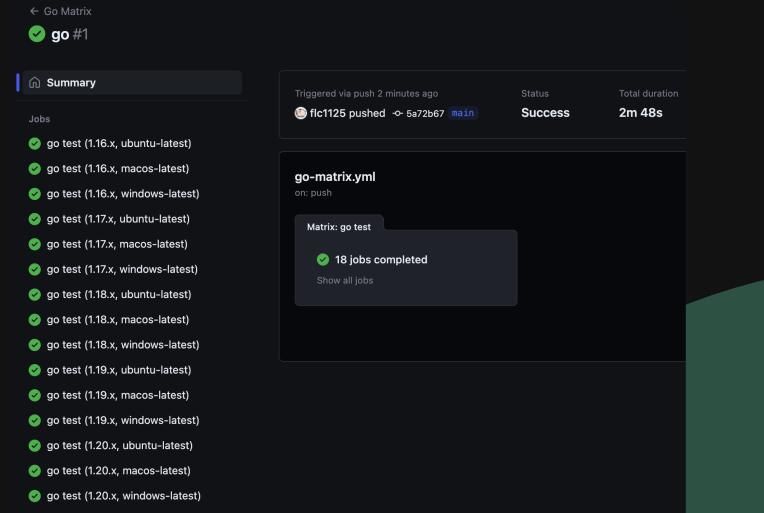
- 矩阵任务
- 依赖和条件



# 高级特性 矩阵任务

Q: 测试用例希望在多个环境，多个不同版本的语言环境下运行？

```
name: Go Test
on:
  push:
    branches: [ main ]
    paths:
      - "go/**"
  env:
    GOPROXY: "https://proxy.golang.org"
  jobs:
    test:
      name: "go test"
      strategy:
        matrix:
          go-version: [ 1.16.x, 1.17.x, 1.18.x, 1.19.x, 1.20.x ]
          platform: [ ubuntu-latest, macos-latest, windows-latest ]
      runs-on: ${{ matrix.platform }}
```



说明：

- 任务运行：并行运行
- 案例地址

# 高级特性

## 依赖和条件

### 依赖

```
name: Test Jobs

on:
  workflow_dispatch:
    inputs:
      test-jobs:
        description: '测试 Jobs'

jobs:
  jobs-1:
    runs-on: ubuntu-latest
    steps:
      - name: Run a one-line script
        run: echo Hello, world! This is jobs-1

  jobs-2:
    runs-on: ubuntu-latest
    needs: # 依赖设定
      - jobs-1
    steps:
      - name: Run a one-line script
        run: echo Hello, world! This is jobs-3, but after jobs-1
```

### 条件

```
name: Test IF

on:
  workflow_dispatch:
    inputs:
      test-jobs:
        description: '测试 IF'

env:
  ENVIRONMENT: prod

jobs:
  dev:
    runs-on: ubuntu-latest
    steps:
      - run: echo The dev env is ${{ env.ENVIRONMENT }}
        if: env.ENVIRONMENT == 'dev'

      - run: echo The prod env is ${{ env.ENVIRONMENT }}
        if: env.ENVIRONMENT == 'prod'
```

```
dev
succeeded 2 minutes ago in 3s

> ✓ Set up job
∅ Run echo The dev env is prod
▼ ✓ Run echo The prod env is prod
  1 ► Run echo The prod env is prod
  6 The prod env is prod
> ✓ Complete job
```

说明：

- 表达式：<https://docs.github.com/zh/actions/learn-github-actions/expressions>
- 您需要使用特定语法指示 GitHub 对表达式求值，而不是将其视为字符串。`${{ <expression> }}`  
在 `if` 条件下使用表达式时，可以省略 `${{ }}` 表达式语法，因为 GitHub Actions 会自动将 `if` 条件作为表达式求值。使用 `${{ }}` 表达式语法将内容转换为字符串，并且字符串是真值。例如，`if: true && ${{ false }}` 的计算结果为 `true`。

# 最佳实践

- 使用徽章来标记 CI 状态
- 使用缓存提高性能
- GITHUB\_TOKEN
- 容器化服务
- 版本控制和代码审查
- 效率和资源管理



# 最佳实践 使用徽章来标记 CI 状态

The screenshot shows a GitHub repository's README.md page. At the top left, there is a logo for "Youdu Go SDK" featuring a blue briefcase icon and a teal owl-like character. Below the logo, the text "Youdu Go SDK" is displayed. A red arrow points from the text "Youdu Go SDK" down to the "Lint" status badge. The badge is green with white text, indicating "passing". To the right of the badge, there is a "Re-run all jobs" button and a three-dot menu button. A dropdown menu is open, showing options: "Create status badge" (which is highlighted in blue), "Search logs", and "Delete all logs". Below the badge, there is a snippet of YAML code:

```
![/](https://github.com/addcnos/youdu/actions/workflows/lint.yml/badge.svg)](https://github.com/addcnos/youdu/actions/workflows/lint.yml)
```

At the bottom left, there is a "Lint" button with a magnifying glass icon.

# 最佳实践 使用缓存提高性能

## 基础版本

```
name: Cache
on:
  push:
    branches: [ main ]

jobs:
  test:
    runs-on: ubuntu-latest

  steps:
    - name: Mkdir
      run: mkdir -p ~/test

    - uses: actions/cache@v3 # 缓存服务
      id: cache-test # 缓存 ID
      with:
        path: ~/test/ # 需要缓存的路径
        key: ${{ runner.os }}-cache-test # 缓存唯一标识
        restore-keys:
          ${{ runner.os }}-cache-
          # 备选标识 (多个换行)

    - run: sleep 10 && echo "Hello world" > ~/test/1.log
```

案例地址

## 首次（未命中缓存）

```
test
succeeded 7 minutes ago in 12s

> ✓ Set up job
> ✓ Run mkdir -p ~/test
✓ Run actions/cache@v3
  1 ► Run actions/cache@v3
  10 Cache not found for input keys: Linux-cache-test, Linux-cache-
  11 /usr/bin/tar -xzf /home/runner/work/_temp/ec1ea60c-fa4f-4b3b-8431-cc07e4f041ba/cache.tzst
  12 Cache restored successfully
  13 Cache restored from key: Linux-cache-test

> ✓ Run sleep 5 && echo "Hello world" > ~/test/1.log
> ✓ Run sleep 5 && echo "Hello world" > ~/test/2.log
✓ Run cat ~/test/*.log
  1 ► Run cat ~/test/*.log
  4 Hello world
  5 Hello world

> ✓ Post Run actions/cache@v3
> ✓ Complete job
```

## 第二次（命中缓存）

```
test
succeeded 12 minutes ago in 6s

> ✓ Set up job
> ✓ Mkdir
✓ Run actions/cache@v3
  1 ► Run actions/cache@v3
  10 Cache Size: ~0 MB (289 B)
  11 /usr/bin/tar -xzf /home/runner/work/_temp/ec1ea60c-fa4f-4b3b-8431-cc07e4f041ba/cache.tzst
  12 Cache restored successfully
  13 Cache restored from key: Linux-cache-test

  (O) Run sleep 5 && echo "Hello world" > ~/test/1.log
  (O) Run sleep 5 && echo "Hello world" > ~/test/2.log

✓ Run cat ~/test/*.log
  1 ► Run cat ~/test/*.log
  4 Hello world
  5 Hello world

> ✓ Post Run actions/cache@v3
> ✓ Complete job
```

9 workflow runs	
✓ init	Cache #9: Commit b6cd664 pushed by flc1125 main
✓ init	Cache #8: Commit 4af790d pushed by flc1125 main

Event ▾ Status ▾ Branch ▾ Actor ▾

命中

未命中

15 minutes ago 15s

16 minutes ago 20s

## 一些第三方服务直接支持的缓存参考

包管理器

用于缓存的 setup-\* 操作

npm、Yarn、pnpm

setup-node

pip、pipenv、Poetry

setup-python

Go `go.sum`

setup-go

PHP `composer.lock`

setup-php

.....

一般情况下，我们会结合上下文等实现缓存 key 的命名。如：

```
$${{ runner.os }}-build-$${{ env.cache-name }}-$${{ hashFiles('**/package-lock.json') }}
```

## 一个简单的 🍄

```
steps:  
  - uses: actions/checkout@v4  
  - uses: actions/setup-go@v4  
    with:  
      go-version: '1.21'  
      check-latest: true  
      cache-dependency-path: /  
        subdir/go.sum  
        tools/go.sum  
    # cache-dependency-path: "**/*.sum"  
  
  - run: go run hello.go
```

如果你想手动清理缓存：

The screenshot shows the GitHub Actions interface. On the left, there's a sidebar with 'Actions' at the top, followed by a list of actions: All workflows, Cache, CI, Cron, Go Matrix, Go Test, Redis, Test Env, Test IF, and Test Jobs. Below this is a 'Management' section with 'Caches' (marked with a red circle containing '1'), 'Deployments', and 'Runners'. A 'Beta' link is also present. The main area is titled 'Caches' with the sub-instruction 'Showing caches from all workflows. Learn more about managing caches.' It displays '2 caches' and a single entry: 'Linux-cache-test' (marked with a red circle containing '2'). This entry includes details: 'main' branch, '289 Bytes cached 27 minutes ago', and 'Last used 27 minutes ago'. To the right of these details is a trash can icon with a red circle containing '3', indicating it can be deleted.

其他小知识：

- 存储库中的多个工作流运行可以**共享缓存**。可以从同一存储库和分支的另一个工作流运行访问和还原为工作流运行中的分支创建的缓存。

# 最佳实践

## GITHUB\_TOKEN

### 💡 自动令牌身份验证

在每个工作流作业开始时，GitHub 会自动创建唯一的 `GITHUB_TOKEN` 机密以在工作流中使用。可以使用 `GITHUB_TOKEN` 在工作流作业中进行身份验证。

那可以做什么？

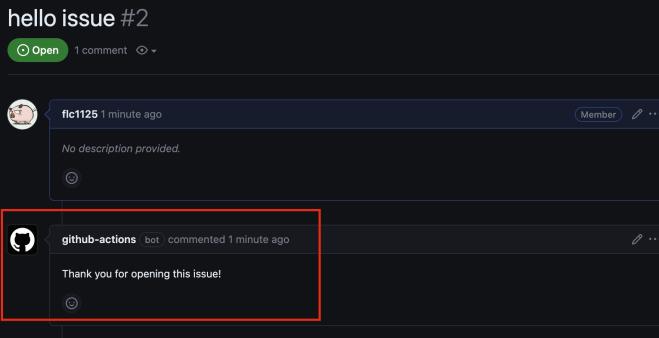
- Github Cli 命令
- 调用 Github REST API
- 仓库写入，生成 `gh-page`
- .....
- 代码自动审查（发起 PR）
- 接入 AI 呢？
- .....



一个

```
name: issue welcome
on:
  issues:
    types:
      - opened

jobs:
  comment:
    runs-on: ubuntu-latest
    permissions: # 设置权限
      issues: write # 写入权限
    steps:
      - run: gh issue comment $ISSUE --body "Thank you for opening this issue!"
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          ISSUE: ${{ github.event.issue.html_url }}
```



案例地址

# 最佳实践 容器化服务

```
name: Redis
on: push

jobs:
  runner-job:
    runs-on: ubuntu-latest

  services:
    redis:
      image: redis # Docker Hub 镜像名称
      ports:
        - 6379:6379

  steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Install Go
      uses: actions/setup-go@v3
    - run: go version

    - name: Run main.go
      run: cd redis && go run main.go
```

```
package main

import (
    "context"
    "time"

    redis "github.com/redis/go-redis/v9"
)

var ctx = context.Background()

func main() {
    rdb := redis.NewClient(&redis.Options{
        Addr: "localhost:6379",
    })

    if err := rdb.Set(ctx, "test", time.Now().String(), time.S
        panic(err)
    }

    if result := rdb.Get(ctx, "test"); result.Err() != nil {
        panic(result.Err())
    } else {
        println(result.Val())
    }
}
```

runner-job  
succeeded 7 minutes ago in 32s

```
> ✓ Set up job
✓ Initialize containers 1
  1 ► Checking docker version
  8 ► Clean up resources from previous jobs
 11 ► Create local container network
 14 ► Starting redis service container
 56 ► Waiting for all services to be ready

> ✓ Checkout code
> ✓ Install Go
> ✓ Run go version
✓ Run main.go
  1 ► Run cd redis && go run main.go
  4 go: downloading github.com/redis/go-redis/v9 v9.1.0
  5 go: downloading github.com/cespare/xxhash/v2 v2.2.0
  6 go: downloading github.com/dgryski/go-rendezvous v0.0.0-20200823014737-9f7001d12a5f
  7 2023-09-17 12:50:06.478041392 +0000 UTC m=+0.000256602 2

> ✓ Post Install Go
> ✓ Post Checkout code
> ✓ Stop containers
> ✓ Complete job
```

说明：

- 容器化服务仅限于在 `ubuntu` 系统下运行
- [案例地址](#)

# 案例分享

- 翻译助手
- 节假日
- 五子棋
- 跨仓库使用
- uptime
- 丰富个人主页
- 单测覆盖率

# 案例分享 翻译助手

基于 issue 的内容，识别中文，自动翻译成英文。

```
name: 'issue-translator'
on:
  issue_comment:
    types: [created]
  issues:
    types: [opened]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: usthe/issues-translate-action@v2.7
        with:
          IS MODIFY TITLE: true
          CUSTOM_BOT_NOTE: Bot detected the issue body's lang
          BOT_GITHUB_TOKEN: ${{ secrets.BOT_GITHUB_TOKEN }}
```

guihouchang on Aug 15Contributor ...

可以尝试在ResponseEncoder这里处理

kratos-ci-bot on Aug 15Collaborator ...

Bot detected the issue body's language is not English, translate it automatically.

You can try to handle it here in ResponseEncoder

# 案例分享 节假日

基于计划任务生成中国节假日信息。

```
name: CI

on:
  push:
    branches: [ master ]
  pull_request:
  workflow_dispatch:
    schedule:
      - cron: "0 12 * * *"
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-python@v4
        with:
          python-version: '3.8'
      - name: Install dependencies
        run: pip install -r dev-requirements.txt
      - name: Setup git user
        run:
          git config user.name "GitHub Actions"
```

```
{
  "$schema": "https://raw.githubusercontent.com/NateScarlet/holiday-data/2023/paper-schema.json",
  "$id": "https://raw.githubusercontent.com/NateScarlet/holiday-data/2023/paper-data.json",
  "year": 2023,
  "papers": [
    {
      "url": "http://www.gov.cn/zhengce/zhengceku/2022-12/08/content_562444.htm"
    }
  ],
  "days": [
    {
      "name": "元旦",
      "date": "2022-12-31",
      "isOffDay": true
    },
    {
      "name": "元旦",
      "date": "2023-01-01",
      "isOffDay": true
    },
    {
      "name": "元旦",
      "date": "2023-01-02",
      "isOffDay": true
    }
  ]
}
```

# 案例分享 五子棋

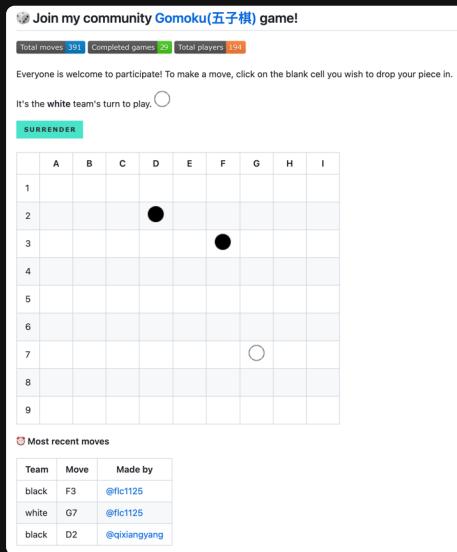
基于创建 issue 触发 Github Actions 更新 README.md

```
name: "Gomoku"

on:
  issues:
    types: [opened]

jobs:
  move:
    runs-on: ubuntu-latest
    if: startsWith(github.event.issue.title, 'gomoku|')
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: Setup Python
        uses: actions/setup-python@v2
        with:
          python-version: 3.8
      - name: Cache PIP
        uses: actions/cache@v2
        with:
          path: ~/.cache/pip
```



# 案例分享 跨仓库使用

Laravel 的 workflows , 统一托管到 [github](#) 下维护。

The screenshot shows the GitHub repository for Laravel's workflow files. The repository name is `laravel/.github`. It contains one branch named `main` and no tags. There is one commit from `stefanzweifel` titled "Abort update-changelog workflow if target branch does not..." dated last month. The repository also includes a `.github/workflows` folder with a workflow file, a `profile` folder, and a `README.md` file.

**README.md**

**Laravel - Community Health Files**

This repository contains the default [community health files](#) for the `laravel` organization.

## laravel/framework

```
name: pull requests

on:
  pull_request_target:
    types: [opened]

permissions:
  pull-requests: write

jobs:
  uneditable:
    uses: laravel/.github/workflows/pull-requests.yml
```

# 案例分享 uptime

定时请求网站状态，并可视化相关数据。[案例地址](#)、[案例仓库地址](#)

 **Flc Status**

Status      Blog      Docs      GitHub

---

## Flc Status is the uptime monitor and status page.

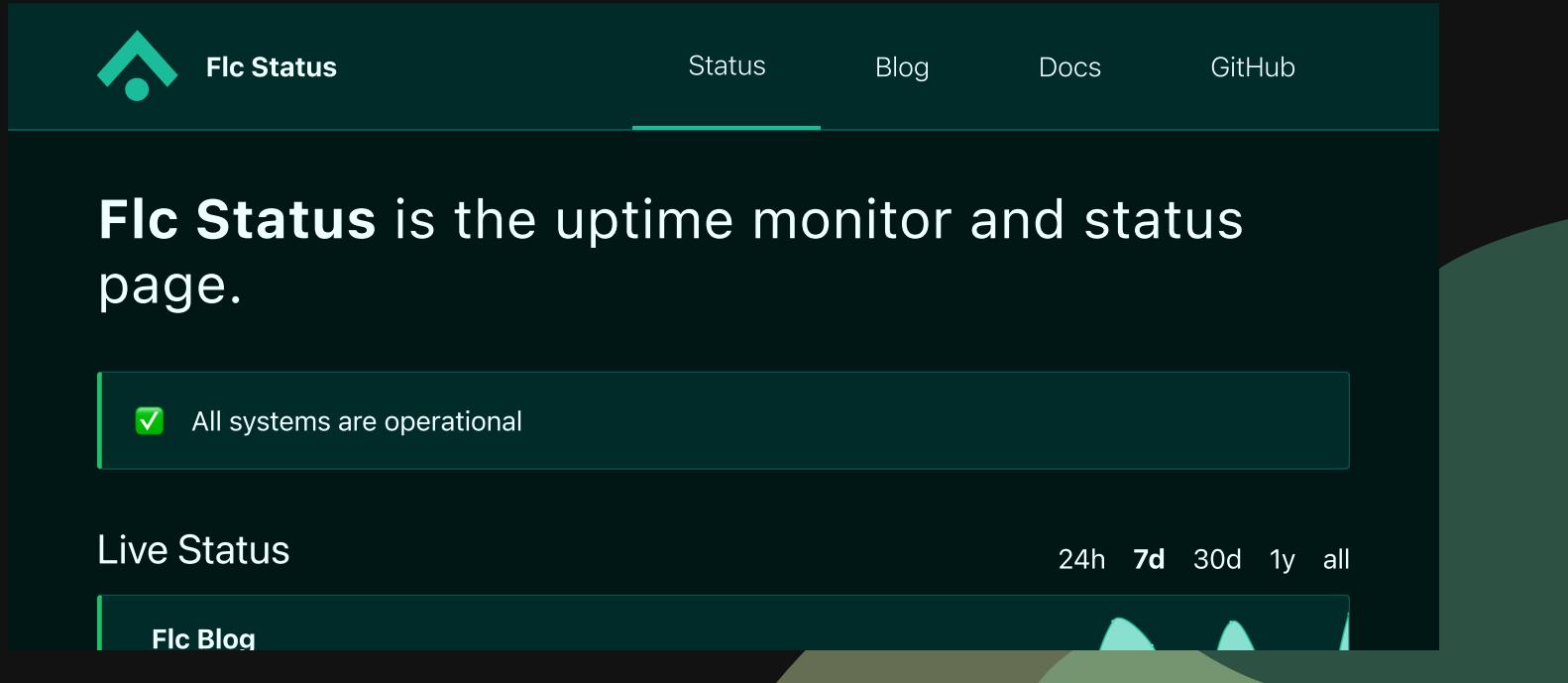
All systems are operational

---

### Live Status

24h 7d 30d 1y all

**Flc Blog**



# 案例分享

## 单测覆盖率

推送代码后，触发单测执行，并上报至第三方平台，便于分析查看。[案例地址](#)

The screenshot shows a GitHub pull request interface with a codecov integration overlay. The pull request is titled "feat: added env context #13" and has been merged. The codecov report indicates 68% coverage, with a green bar chart showing a 6.55% increase from master to the pull request. The report details the following metrics:

	master	#13	Diff
Coverage	65.27%	66.32%	+0.55%
Files	8	17	+9
Lines	488	496	+8
Hits	321	329	+8
Misses	156	156	0
Partials	11	11	0

The report also lists files changed with their respective coverage changes:

Files Changed	Coverage Δ
cacheManager.go	+0.00% <0.00% (e)
cachedRedisStore.go	+5.97% >0.00% (g)
cacheRepository.go	+0.00% <0.00% (e)
envContext.go	+0.00% <0.00% (e)
envProvider.go	+0.00% <0.00% (e)
serialZerion.go	+3.33% <0.00% (e)

A note at the bottom of the report states: "We're building smart automated test selection to slash your CI/CD build times. Learn more".

```
name: Go Test
on:
  push:
    branches: [ master, feature/* ]
  pull_request:
    branches: [ master ]
  env:
    GOPROXY: "https://proxy.golang.org"
  jobs:
    test:
      name: "go test"
      strategy:
        matrix:
          go-version: [ 1.18.x, 1.19.x, 1.20.x, 1.21.x ]
          platform: [ ubuntu-latest ]
      runs-on: ${{ matrix.platform }}
    services:
```

