

# GitHub Actions 快速参考

工作流、触发器、作业、密钥、缓存、产物

## 工作流基础

### 最简工作流

```
# .github/workflows/ci.yml
name: CI
on: push
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: echo "Hello from CI"
```

### 核心概念

<b>Workflow</b>	<code>.github/workflows/</code> 中定义自动化的 YAML 文件
<b>Event</b>	触发工作流的事件 (push、PR、定时等)
<b>Job</b>	在同一 Runner 上运行的一组步骤
<b>Step</b>	单个任务——运行命令或使用 action
<b>Runner</b>	执行作业的虚拟机 ( <code>ubuntu-latest</code> 、 <code>macos-latest</code> 等)
<b>Action</b>	通过 <code>uses:</code> 引用的可复用代码单元

## 触发器

### 常用事件

```
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
  schedule:
    - cron: "0 6 * * 1" # every Monday 6 AM UTC
  workflow_dispatch: # manual trigger
```

### 事件过滤器

<b>branches:</b>	仅对特定分支触发
<b>paths:</b>	仅在匹配文件变更时触发
<b>tags:</b>	在 tag push 时触发 ( <b>v*</b> )
<b>types: [opened, synchronize]</b>	过滤 PR 活动类型
<b>branches-ignore:</b>	排除特定分支
<b>paths-ignore:</b>	排除特定文件路径

## 作业与步骤

### 作业配置

```
jobs:
  test:
    runs-on: ubuntu-latest
    needs: build # depends on build job
    if: github.ref == 'refs/heads/main'
    timeout-minutes: 10
    steps:
      - uses: actions/checkout@v4
      - run: npm test
```

### 步骤类型

<b>run:</b>	执行 shell 命令
<b>uses:</b>	使用发布的 action
<b>with:</b>	向 action 传入参数
<b>name:</b>	UI 中显示的名称
<b>id:</b>	通过 <code>steps.&lt;id&gt;.outputs</code> 引用步骤输出
<b>if:</b>	条件执行
<b>continue-on-error: true</b>	步骤失败时不中止作业

## Actions

### 使用 Actions

```
steps:
  - uses: actions/checkout@v4
  - uses: actions/setup-node@v4
    with:
      node-version: 20
  - uses: ./github/actions/my-action # local action
```

### 常用 Actions

<b>actions/checkout@v4</b>	检出仓库代码
<b>actions/setup-node@v4</b>	安装 Node.js
<b>actions/setup-python@v5</b>	安装 Python
<b>actions/upload-artifact@v4</b>	上传构建产物
<b>actions/download-artifact@v4</b>	从其他作业下载产物
<b>actions/cache@v4</b>	跨运行缓存依赖
<b>actions/github-script@v7</b>	使用 GitHub API 客户端运行 JS

## 环境变量

### 设置变量

```
env: # workflow-level
  NODE_ENV: production
jobs:
  build:
    env: # job-level
      CI: true
    steps:
      - run: echo "$MY_VAR" # step-level
        env:
          MY_VAR: hello
```

### 默认变量

<b>github.sha</b>	触发工作流的 commit SHA
<b>github.ref</b>	分支或 tag 引用 ( <b>refs/heads/main</b> )
<b>github.repository</b>	Owner/repo 名称
<b>github.actor</b>	触发工作流的用户
<b>github.event_name</b>	触发工作流的事件名
<b>runner.os</b>	Runner 操作系统 ( <b>Linux</b> 、 <b>macOS</b> 、 <b>Windows</b> )

## 密钥

### 使用密钥

```
steps:
  - run: deploy --token "$TOKEN"
    env:
      TOKEN: ${ secrets.DEPLOY_TOKEN }
  - uses: some/action@v1
    with:
      api-key: ${ secrets.API_KEY }
```

### 密钥规则

<b>secrets.GITHUB_TOKEN</b>	自动生成的仓库范围 token
<b>Settings → Secrets</b>	在仓库或组织设置中添加密钥
<b>Masking</b>	密钥值在日志中自动脱敏
<b>Environment secrets</b>	作用域限于部署环境
<b>Org secrets</b>	在组织内跨仓库共享

## 矩阵策略

### 矩阵构建

```
jobs:
  test:
    strategy:
      matrix:
        os: [ubuntu-latest, macos-latest]
        node: [18, 20]
    runs-on: ${ matrix.os }
    steps:
      - uses: actions/setup-node@v4
        with:
          node-version: ${ matrix.node }
```

### 矩阵选项

<b>matrix:</b>	定义要展开的变量组合
<b>include:</b>	添加额外组合到矩阵
<b>exclude:</b>	移除特定组合
<b>fail-fast: false</b>	其中一个失败时继续其他
<b>max-parallel: 2</b>	限制并发矩阵作业数

## 缓存

### 缓存依赖

```
- uses: actions/cache@v4
  with:
    path: ~/.npm
    key: npm-${ hashFiles('package-lock.json') }
    restore-keys: npm-
```

### 内置缓存

```
- uses: actions/setup-node@v4
  with:
    node-version: 20
    cache: npm # auto-cache for npm/yarn/pnpm
- uses: actions/setup-python@v5
  with:
    python-version: "3.12"
    cache: pip # auto-cache for pip
```

## 产物

### 上传与下载

```
- uses: actions/upload-artifact@v4
  with:
    name: build-output
    path: dist/
    retention-days: 7
- uses: actions/download-artifact@v4
  with:
    name: build-output
```

### 产物说明

<b>retention-days</b>	N 天后自动删除 (默认 90)
<b>path</b>	要上传的文件或目录 (支持 glob)
<b>Cross-job</b>	一个作业上传, 另一个用 <code>needs:</code> 下载
<b>compression-level</b>	0 (不压缩) 到 9 (最大), 默认 6

## 常用模式

### 条件部署

```
- name: Deploy to production
  if: github.ref == 'refs/heads/main'
  run: ./deploy.sh
- name: Post PR comment
  if: github.event_name == 'pull_request'
  run: gh pr comment $PR --body "Build passed"
```

# GitHub Actions 快速参考

---

## 常用表达式

<code>success()</code>	所有前序步骤通过时为真
<code>failure()</code>	任一前序步骤失败时为真
<code>always()</code>	无论状态如何都运行（清理）
<code>cancelled()</code>	工作流被取消时为真
<code>contains(github.ref, 'release')</code>	字符串包含检查
<code>startsWith(github.ref, 'refs/tags')</code>	字符串前缀检查
<code>hashFiles('**/lock*')</code>	文件的 SHA-256（用于缓存 key）