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

```yaml
# .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

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

触发器

常用事件

```yaml
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

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

作业与步骤

作业配置

```yaml
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

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

Actions

使用 Actions

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

常用 Actions

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

环境变量

设置变量

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

默认变量

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

密钥

使用密钥

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

密钥规则

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

矩阵策略

矩阵构建

```yaml
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

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

缓存

缓存依赖

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

内置缓存

```yaml
- 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

```yaml
- 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

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

常用模式

条件部署

```yaml
- 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

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