REFERÊNCIA RÁPIDA GITHUB ACTIONS

Workflows, triggers, jobs, secrets, cache, artifacts

Fundamentos de Workflow

Workflow Mínimo

```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"
```

Conceitos Principais

`Workflow`	Arquivo YAML em `.github/workflows/` que define automação
`Event`	Gatilho que inicia um workflow (push, PR, schedule, etc.)
`Job`	Conjunto de steps que rodam no mesmo runner
`Step`	Tarefa individual — executa comando ou usa uma action
`Runner`	VM que executa jobs (`ubuntu-latest`, `macos-latest`, `windows-latest`)
`Action`	Unidade de código reutilizável referenciada com `uses:`

Triggers

Eventos Comuns

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

Filtros de Evento

`branches:`	Acionar somente para branches específicas
`paths:`	Acionar somente quando arquivos correspondentes mudam
`tags:`	Acionar em pushes de tag (`v*`)
`types: [opened, synchronize]`	Filtrar tipos de atividade de PR
`branches-ignore:`	Excluir branches específicas
`paths-ignore:`	Excluir caminhos de arquivo específicos

Jobs e Steps

Configuração de Job

```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
```

Tipos de Step

`run:`	Executar um comando shell
`uses:`	Usar uma action publicada
`with:`	Passar inputs para uma action
`name:`	Nome de exibição na UI
`id:`	Referenciar outputs do step via `steps.<id>.outputs`
`if:`	Execução condicional
`continue-on-error: true`	Não falhar o job se o step falhar

Actions

Usando Actions

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

Actions Populares

`actions/checkout@v4`	Fazer checkout do código do repositório
`actions/setup-node@v4`	Instalar Node.js
`actions/setup-python@v5`	Instalar Python
`actions/upload-artifact@v4`	Fazer upload de artefatos de build
`actions/download-artifact@v4`	Baixar artefatos de outro job
`actions/cache@v4`	Cachear dependências entre execuções
`actions/github-script@v7`	Executar JS com cliente da API do GitHub

Variáveis de Ambiente

Definindo Variáveis

```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
```

Variáveis Padrão

`github.sha`	SHA do commit que acionou o workflow
`github.ref`	Ref de branch ou tag (`refs/heads/main`)
`github.repository`	Nome owner/repo
`github.actor`	Usuário que acionou o workflow
`github.event_name`	Evento que acionou o workflow
`runner.os`	SO do runner (`Linux`, `macOS`, `Windows`)

Secrets

Usando Secrets

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

Regras de Secrets

`secrets.GITHUB_TOKEN`	Token gerado automaticamente com escopo do repositório
`Settings → Secrets`	Adicionar secrets nas configurações do repo ou org
`Masking`	Valores de secrets são mascarados nos logs automaticamente
`Environment secrets`	Com escopo para um ambiente de deploy
`Org secrets`	Compartilhados entre repos de uma organização

Estratégia Matrix

Builds em Matrix

```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 }}
```

Opções de Matrix

`matrix:`	Definir combinações de variáveis para expandir
`include:`	Adicionar combinações extras à matrix
`exclude:`	Remover combinações específicas
`fail-fast: false`	Continuar outros jobs se um falhar
`max-parallel: 2`	Limitar jobs matrix concorrentes

Cache

Cachear Dependências

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

Cache Embutido

```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
```

Artifacts

Upload e Download

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

Notas sobre Artifacts

`retention-days`	Auto-excluir após N dias (padrão 90)
`path`	Arquivo ou diretório para upload (suporta globs)
`Cross-job`	Upload em um job, download em outro com `needs:`
`compression-level`	0 (nenhuma) a 9 (máxima), padrão 6

Padrões Comuns

Deploy Condicional

```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"
```

Expressões Úteis

`success()`	Verdadeiro se todos os steps anteriores passaram
`failure()`	Verdadeiro se algum step anterior falhou
`always()`	Executar independente do status (limpeza)
`cancelled()`	Verdadeiro se o workflow foi cancelado
`contains(github.ref, 'release')`	Verificação se string contém
`startsWith(github.ref, 'refs/tags')`	Verificação de prefixo de string
`hashFiles('**/lock*')`	SHA-256 de arquivos (para chaves de cache)