RIFERIMENTO RAPIDO GITHUB ACTIONS

Workflow, trigger, job, segreti, caching, artifact

Basi del Workflow

Workflow Minimale

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

Concetti Chiave

`Workflow`	File YAML in `.github/workflows/` che definisce l'automazione
`Event`	Trigger che avvia un workflow (push, PR, pianificazione, ecc.)
`Job`	Set di step che girano sullo stesso runner
`Step`	Task individuale — esegue un comando o usa un'action
`Runner`	VM che esegue i job (`ubuntu-latest`, `macos-latest`, `windows-latest`)
`Action`	Unità di codice riutilizzabile referenziata con `uses:`

Trigger

Eventi Comuni

```
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
  schedule:
    - cron: "0 6 * * 1"    # ogni lunedì alle 6 AM UTC
  workflow_dispatch:          # trigger manuale
```

Filtri Evento

`branches:`	Trigger solo per branch specifici
`paths:`	Trigger solo quando i file corrispondenti cambiano
`tags:`	Trigger su push di tag (`v*`)
`types: [opened, synchronize]`	Filtra tipi di attività PR
`branches-ignore:`	Esclude branch specifici
`paths-ignore:`	Esclude percorsi file specifici

Job e Step

Configurazione Job

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

Tipi di Step

`run:`	Esegue un comando shell
`uses:`	Usa un'action pubblicata
`with:`	Passa input a un'action
`name:`	Nome visualizzato nell'interfaccia
`id:`	Riferimento output step via `steps.<id>.outputs`
`if:`	Esecuzione condizionale
`continue-on-error: true`	Non fallire il job se lo step fallisce

Action

Usare le Action

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

Action Popolari

`actions/checkout@v4`	Clona il codice del repository
`actions/setup-node@v4`	Installa Node.js
`actions/setup-python@v5`	Installa Python
`actions/upload-artifact@v4`	Carica artifact di build
`actions/download-artifact@v4`	Scarica artifact da un altro job
`actions/cache@v4`	Mette in cache le dipendenze tra le esecuzioni
`actions/github-script@v7`	Esegue JS con il client API GitHub

Variabili d'Ambiente

Impostare Variabili

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

Variabili Predefinite

`github.sha`	SHA del commit che ha avviato il workflow
`github.ref`	Ref del branch o tag (`refs/heads/main`)
`github.repository`	Nome owner/repo
`github.actor`	Utente che ha avviato il workflow
`github.event_name`	Evento che ha avviato il workflow
`runner.os`	OS del runner (`Linux`, `macOS`, `Windows`)

Segreti

Usare i Segreti

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

Regole Segreti

`secrets.GITHUB_TOKEN`	Token auto-generato con scope al repository
`Settings → Secrets`	Aggiungi segreti nelle impostazioni repo o org
`Mascheramento`	I valori segreti vengono mascherati automaticamente nei log
`Segreti ambiente`	Con scope a un ambiente di deployment
`Segreti organizzazione`	Condivisi tra i repo nell'organizzazione

Matrix Strategy

Build a Matrix

```
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 }}
```

Opzioni Matrix

`matrix:`	Definisce combinazioni di variabili da espandere
`include:`	Aggiunge combinazioni extra alla matrix
`exclude:`	Rimuove combinazioni specifiche
`fail-fast: false`	Continua altri job se uno fallisce
`max-parallel: 2`	Limita i job matrix concorrenti

Caching

Cache delle Dipendenze

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

Caching Integrato

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

Artifact

Upload e Download

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

Note sugli Artifact

`retention-days`	Eliminazione automatica dopo N giorni (default 90)
`path`	File o directory da caricare (supporta glob)
`Cross-job`	Carica in un job, scarica in un altro con `needs:`
`compression-level`	Da 0 (nessuna) a 9 (massima), default 6

Pattern Comuni

Deployment Condizionale

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

Espressioni Utili

`success()`	True se tutti gli step precedenti sono passati
`failure()`	True se uno step precedente è fallito
`always()`	Esegui indipendentemente dallo stato (cleanup)
`cancelled()`	True se il workflow è stato annullato
`contains(github.ref, 'release')`	Controllo contiene stringa
`startsWith(github.ref, 'refs/tags')`	Controllo prefisso stringa
`hashFiles('**/lock*')`	SHA-256 dei file (per chiavi cache)