Referencia Rápida de GitHub Actions

Workflows, triggers, jobs, secretos, caché, artefactos

Fundamentos del workflow

Workflow mínimo

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

Conceptos clave

Workflow	Archivo YAML en **.github/workflows/** que define la automatización
Event	Trigger que inicia un workflow (push, PR, schedule, etc.)
Job	Conjunto de steps que se ejecutan en el mismo runner
Step	Tarea individual — ejecuta un comando o usa una action
Runner	VM que ejecuta los jobs (**ubuntu-latest**, **macos-latest**, **windows-latest**)
Action	Unidad de código reutilizable referenciada con **uses:**

Triggers

Eventos comunes

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

Filtros de eventos

branches:	Activar solo para ramas específicas
paths:	Activar solo cuando cambian archivos coincidentes
tags:	Activar en push de tags (**v***)
types: [opened, synchronize]	Filtrar tipos de actividad de PR
branches-ignore:	Excluir ramas específicas
paths-ignore:	Excluir rutas de archivo específicas

Jobs y Steps

Configuración de job

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

Tipos de step

run:	Ejecutar un comando shell
uses:	Usar una action publicada
with:	Pasar entradas a una action
name:	Nombre visible en la interfaz
id:	Referenciar salidas del step con **steps.<id>.outputs**
if:	Ejecución condicional
continue-on-error: true	No fallar el job si falla el step

Actions

Usar actions

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

Actions populares

actions/checkout@v4	Obtener código del repositorio
actions/setup-node@v4	Instalar Node.js
actions/setup-python@v5	Instalar Python
actions/upload-artifact@v4	Subir artefactos del build
actions/download-artifact@v4	Descargar artefactos de otro job
actions/cache@v4	Cachear dependencias entre ejecuciones
actions/github-script@v7	Ejecutar JS con el cliente de la API de GitHub

Variables de entorno

Establecer variables

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

Variables predeterminadas

github.sha	SHA del commit que activó el workflow
github.ref	Ref de rama o tag (**refs/heads/main**)
github.repository	Nombre owner/repo
github.actor	Usuario que activó el workflow
github.event_name	Evento que activó el workflow
runner.os	OS del runner (**Linux**, **macOS**, **Windows**)

Secretos

Usar secretos

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

Reglas de secretos

secrets.GITHUB_TOKEN	Token autogenerado con scope del repo
Settings → Secrets	Agregar secretos en la configuración del repo u org
Masking	Los valores de secretos se enmascaran en los logs automáticamente
Environment secrets	Con scope a un entorno de despliegue
Org secrets	Compartidos entre repos de una organización

Estrategia de matriz

Builds en matriz

```
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 }}
```

Opciones de matriz

matrix:	Definir combinaciones de variables a expandir
include:	Agregar combinaciones adicionales a la matriz
exclude:	Eliminar combinaciones específicas
fail-fast: false	Continuar otros jobs si uno falla
max-parallel: 2	Limitar jobs de matriz concurrentes

Caché

Cachear dependencias

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

Caché integrado

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

Artefactos

Subir y descargar

```
- 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 artefactos

retention-days	Auto-eliminar después de N días (por defecto 90)
path	Archivo o directorio a subir (admite globs)
Cross-job	Subir en un job, descargar en otro con **needs:**
compression-level	0 (ninguna) a 9 (máxima), por defecto 6

Patrones comunes

Despliegue condicional

```
- 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"
```

Referencia Rápida de GitHub Actions

Expresiones útiles

`success()`	Verdadero si todos los steps anteriores pasaron
`failure()`	Verdadero si algún step anterior falló
`always()`	Ejecutar sin importar el estado (limpieza)
`cancelled()`	Verdadero si el workflow fue cancelado
`contains(github.ref, 'release')`	Verificar si la cadena contiene
`startsWith(github.ref, 'refs/tags')`	Verificar prefijo de cadena
`hashFiles('**/lock*')`	SHA-256 de archivos (para claves de caché)