

Contents

Cheatsheet GitHub Actions	1
Anatomie d'un workflow	1
Déclencheurs (on:)	2
Jobs : dépendances, conditions, services	2
Expressions et contextes	3
Secrets, variables, GITHUB_TOKEN	3
Actions courantes	4
Lire un échec	5
Encadré : pièges fréquents	5

Cheatsheet GitHub Actions

Un **workflow** (fichier YAML dans `.github/workflows/`) est déclenché par un **événement** ; il contient des **jobs** (en parallèle par défaut) ; chaque job tourne sur un **runner** et enchaîne des **steps** (une commande `run:` ou une action `uses:`).

Anatomie d'un workflow

```
name: CI                                # nom affiché dans l'onglet Actions

on:                                     # QUAND ?
  push:
    branches: [main]
  pull_request:

permissions:                            # droits du GITHUB_TOKEN (au plus juste)
  contents: read

env:                                    # variables pour tout le workflow
  NODE_VERSION: "20"

jobs:                                   # QUOI ?
  test-api:
    runs-on: ubuntu-latest              # OÙ ? (runner hébergé par GitHub)
    defaults:
      run:
        working-directory: api
    steps:
      - uses: actions/checkout@v4        # récupère le code
      - uses: actions/setup-node@v4      # paramètres de l'action
        with:
          node-version: ${ env.NODE_VERSION }
          cache: npm
          cache-dependency-path: api/package-lock.json
      - run: npm ci
      - run: npm run lint
```

- **run:** npm test

Règles YAML : **espaces uniquement** (2 par niveau), pas de tabulation ; - = élément de liste ; clé: valeur (espace après les deux-points) ; | = texte multi-lignes ; les valeurs « bizarres » (20, on, yes) se mettent entre guillemets.

Déclencheurs (on:)

Déclencheur	Quand
push (branches:, tags: ["v*"], paths:) pull_request	Un push sur ces branches/tags/fichiers Ouverture/mise à jour d'une PR (le code de la PR)
workflow_dispatch (+ inputs:)	Bouton « Run workflow » manuel (ex. rollback avec un tag)
schedule: - cron: "0 6 * * 1"	Planifié (UTC) : min heure jour mois jour-semaine
workflow_call	Appelé par un autre workflow
release: types: [published]	Publication d'une release

```
on:  
  workflow_dispatch:  
    inputs:  
      tag:  
        description: "Tag à déployer"  
        required: true  
        default: "latest"  
# usage : ${ inputs.tag }
```

Jobs : dépendances, conditions, services

Mot-clé	Effet
needs: [build] if: github.ref == 'refs/heads/main' if: failure() / always() runs-on: [self-hosted, lab] strategy.matrix	Attend la réussite de build Exécute seulement si... Après un échec / quoi qu'il arrive Runner chez vous (labels) Répète le job avec plusieurs valeurs (ex. versions de Node)
environment: production	Environnement GitHub (secrets dédiés, approbation)
concurrency: { group: deploy, cancel-in-progress: false } timeout-minutes: 10 outputs:	Un seul déploiement à la fois Évite un job bloqué Passe une valeur à un autre job

Service Postgres pour les tests d'intégration :

```

jobs:
  test-api:
    runs-on: ubuntu-latest
    services:
      postgres:
        image: postgres:16-alpine
        env:
          POSTGRES_USER: taskflow
          POSTGRES_PASSWORD: taskflow
          POSTGRES_DB: taskflow
        ports: ["5432:5432"]
        options: >-
          --health-cmd "pg_isready -U taskflow"
          --health-interval 5s --health-timeout 3s --health-retries 10
    env:
      DATABASE_URL: postgres://taskflow:taskflow@localhost:5432/taskflow

```

Expressions et contextes

Syntaxe : `${{ ... }}`.

Expression	Valeur
<code>\${{ github.sha }}</code>	SHA complet du commit
<code>\${{ github.ref_name }}</code>	Nom de la branche ou du tag
<code>\${{ github.actor }}</code>	Qui a déclenché
<code>\${{ github.repository }}</code> / <code>repository_owner</code>	owner/repo / owner
<code>\${{ github.event_name }}</code>	push, pull_request...
<code>\${{ runner.os }}</code>	Linux...
<code>\${{ env.NOM }}</code> / <code>\${{ vars.NOM }}</code> / <code>\${{ secrets.NOM }}</code>	Variable d'env / de dépôt / secret
<code>\${{ needs.build.outputs.tag }}</code>	Sortie d'un autre job
<code>\${{ steps.id.outputs.valeur }}</code>	Sortie d'une étape précédente
<code>contains(x, 'y')</code> , <code>startsWith(x, 'v')</code> , <code>format(...)</code> , <code>hashFiles('*/package-lock.json')</code>	Fonctions
<code>success()</code> , <code>failure()</code> , <code>always()</code> , <code>cancelled()</code>	Statut (dans if:)

SHA court dans un step, à exposer aux suivants :

```

- id: meta
  run: echo "short=${GITHUB_SHA::7}" >> "$GITHUB_OUTPUT"
- run: echo "Version ${ steps.meta.outputs.short }"

```

Secrets, variables, GITHUB_TOKEN

Type	Où le créer	Pour quoi	Lecture
Secret	Settings > Secrets and variables > Actions	Mot de passe, jeton (masqué dans les logs)	<code>{{ secrets.X }}</code>
Variable (vars)	Même page, onglet Variables	Config non secrète (nom d'hôte, port)	<code>{{ vars.X }}</code>
env	Dans le YAML	Valeur locale au workflow/job/step	<code>{{ env.X }}</code> ou <code>\$X</code>
GITHUB_TOKEN	Automatique	Jeton temporaire du workflow (push ghcr.io, commenter une PR)	<code>{{ secrets.GITHUB_TOKEN }}</code>

permissions: borne le GITHUB_TOKEN. Pour publier sur ghcr.io : packages: write (et contents: read).

Actions courantes

Action	Rôle
actions/checkout@v4	Récupère le code du dépôt
actions/setup-node@v4	Installe Node (+ cache: npm)
actions/cache@v4	Cache personnalisé
actions/upload-artifact@v4 / download-artifact@v4	Garde/partage des fichiers entre jobs (rapports, dist/)
docker/setup-buildx-action@v3	Prépare Docker Buildx
docker/login-action@v3	Se connecte au registre
docker/metadata-action@v5	Calcule tags/labels d'image
docker/build-push-action@v6	Construit et pousse l'image

Build + push vers ghcr.io (tags sha et latest) :

```
permissions:
  contents: read
  packages: write
steps:
  - uses: actions/checkout@v4
  - uses: docker/login-action@v3
    with:
      registry: ghcr.io
      username: {{ github.actor }}
      password: {{ secrets.GITHUB_TOKEN }}
  - uses: docker/build-push-action@v6
    with:
      context: ./api
      push: true
      tags: |
```

```
ghcr.io/${{ github.repository_owner }}/taskflow-api:${{ github.sha }}
ghcr.io/${{ github.repository_owner }}/taskflow-api:latest
```

Les noms d'images ghcr.io doivent être en **minuscules**.

Job de déploiement sur le runner du lab :

```
deploy:
  needs: build
  runs-on: [self-hosted, lab]
  environment: production
  steps:
    - uses: actions/checkout@v4
    - run: ./deploy.sh "${GITHUB_SHA:7}"
      working-directory: ${vars.DEPLOY_DIR }
```

Badge de statut (README) : ![CI] (<https://github.com/<owner>/<repo>/actions/workflows/ci.y>)

Lire un échec

1. Onglet **Actions** > le run rouge > le job rouge > l'étape marquée par une croix : ouvrez-la, lisez **les dernières lignes avant l'erreur**.
2. Reproduisez la commande en local (`npm ci && npm test`, dans le bon dossier).
3. Corrigez, committez, poussez : le workflow se relance. *Re-run jobs* relance sans changer le code (utile pour un incident réseau).
4. Activer plus de logs : *Re-run jobs* > *Enable debug logging*.

Encadré : pièges fréquents

- **Indentation / tabulation** : l'erreur « Invalid workflow file » indique la ligne. Utilisez l'extension GitHub Actions de VS Code.
- **Le workflow n'apparaît pas** : le fichier doit être dans `.github/workflows/` (avec le `s`) et en `.yml/.yaml`, sur la branche concernée.
- **Secret vide** : les secrets ne sont **pas** transmis aux workflows déclenchés par des PR venant de forks. Et un nom mal orthographié donne une chaîne vide sans erreur.
- **working-directory** oublié : `npm ci` échoue (« package.json not found ») dans un monorepo `api/` + `front/`.
- **Image en majuscules** sur ghcr.io : refusée. Pensez à `permissions: packages: write`.
- **Tag @main ou sans version** pour une action : non reproductible. Épinglez (@v4).
- **Jamais de secret dans run: echo** : même masqué, il peut fuiter par transformation.
- **Runner self-hosted sur dépôt public** : dangereux (voir `lab/runner/README.md`).
- `needs` : oublié : les jobs partent en parallèle (le déploiement ne doit pas passer avant les tests).
- Cron en **UTC**, et GitHub peut retarder les workflows planifiés.