GitHub Actions 빠른 참조

워크플로우, 트리거, 잡, 시크릿, 캐싱, 아티팩트

워크플로우 기본

최소 워크플로우

```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"
```

핵심 개념

Workflow	자동화를 정의하는 **.github/workflows/**의 YAML 파일
Event	워크플로우를 시작하는 트리거 (push, PR, schedule 등)
Job	동일한 러너에서 실행되는 스텝의 집합
Step	명령 실행 또는 액션 사용하는 개별 작업
Runner	잡을 실행하는 VM (**ubuntu-latest**, **macos-latest**, **windows-latest**)
Action	**uses:**로 참조하는 재사용 가능한 코드 단위

트리거

주요 이벤트

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

이벤트 필터

branches:	특정 브랜치에서만 트리거
paths:	일치하는 파일이 변경될 때만 트리거
tags:	태그 푸시 시 트리거 (**v***)
types: [opened, synchronize]	PR 활동 유형 필터
branches-ignore:	특정 브랜치 제외
paths-ignore:	특정 파일 경로 제외

잡 & 스텝

잡 설정

```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
```

스텝 유형

run:	셸 명령 실행
uses:	게시된 액션 사용
with:	액션에 입력 전달
name:	UI에 표시되는 이름
id:	**steps.<id>.outputs**로 스텝 출력 참조
if:	조건부 실행
continue-on-error: true	스텝 실패 시 잡을 실패하지 않음

액션

액션 사용

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

주요 액션

actions/checkout@v4	저장소 코드 체크아웃
actions/setup-node@v4	Node.js 설치
actions/setup-python@v5	Python 설치
actions/upload-artifact@v4	빌드 아티팩트 업로드
actions/download-artifact@v4	다른 잡에서 아티팩트 다운로드
actions/cache@v4	실행 간 의존성 캐싱
actions/github-script@v7	GitHub API 클라이언트로 JS 실행

환경 변수

변수 설정

```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
```

기본 변수

github.sha	워크플로우를 트리거한 커밋 SHA
github.ref	브랜치 또는 태그 ref (**refs/heads/main**)
github.repository	소유자/저장소 이름
github.actor	워크플로우를 트리거한 사용자
github.event_name	워크플로우를 트리거한 이벤트
runner.os	러너 OS (**Linux**, **macOS**, **Windows**)

시크릿

시크릿 사용

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

시크릿 규칙

secrets.GITHUB_TOKEN	저장소 범위로 자동 생성된 토큰
Settings → Secrets	저장소 또는 조직 설정에서 시크릿 추가
Masking	시크릿 값은 로그에서 자동으로 마스킹
Environment secrets	배포 환경에 범위 지정
Org secrets	조직 내 저장소 간 공유

매트릭스 전략

매트릭스 빌드

```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 }}
```

매트릭스 옵션

matrix:	확장할 변수 조합 정의
include:	매트릭스에 추가 조합 더하기
exclude:	특정 조합 제거
fail-fast: false	하나 실패해도 다른 잡 계속 실행
max-parallel: 2	동시 매트릭스 잡 수 제한

캐싱

의존성 캐시

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

내장 캐싱

```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
```

아티팩트

업로드 & 다운로드

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

아티팩트 메모

retention-days	N일 후 자동 삭제 (기본 90일)
path	업로드할 파일 또는 디렉터리 (글로브 지원)
Cross-job	한 잡에서 업로드, **needs:**로 다른 잡에서 다운로드
compression-level	0(없음)에서 9(최대), 기본값 6

일반 패턴

조건부 배포

```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 빠른 참조

유용한 표현식

`success()`	이전 모든 스텝 통과 시 참
`failure()`	이전 스텝 중 하나라도 실패 시 참
`always()`	상태 관계없이 실행 (정리 작업)
`cancelled()`	워크플로우가 취소된 경우 참
`contains(github.ref, 'release')`	문자열 포함 검사
`startsWith(github.ref, 'refs/tags')`	문자열 접두사 검사
`hashFiles('**/lock*')`	파일의 SHA-256 (캐시 키용)