GitLab CI/CD 빠른 참조

파이프라인, 잡, 스테이지, 변수, 아티팩트, 환경

파이프라인 기본

파이프라인 동작 방식

Pipeline	최상위 컨테이너; 커밋/트리거당 하나
Stage	병렬로 실행되는 잡의 그룹
Job	스테이지 내 단일 작업 (스크립트)
Runner	잡을 실행하는 에이전트

파이프라인 트리거

Push to branch	자동 (기본값)
Merge request	workflow:rules 또는 only: merge_requests
Schedule	프로젝트 설정의 CI/CD → Schedules
API	POST /projects/:id/trigger/pipeline
Manual	CI/CD 메뉴의 Run Pipeline 버튼

.gitlab-ci.yml

최소 설정

```
stages: [build, test, deploy]
build-job:
  stage: build
  script: echo "Compiling..."
```

전역 키워드

stages	스테이지 순서 정의
default	모든 잡의 기본값
variables	전역 CI/CD 변수
workflow	파이프라인 생성 조건 제어
include	외부 YAML 파일 가져오기

템플릿 포함

```
include:
  - template: Auto-DevOps.gitlab-ci.yml
  - local: .ci/lint.yml
  - project: 'group/shared-ci'
    file: '/templates/deploy.yml'
```

잡

잡 정의

```
test-unit:
  stage: test
  image: node:20
  script:
    - npm ci
    - npm test
```

잡 키워드

script	실행할 셸 명령 (필수)
before_script	메인 스크립트 전 명령
after_script	실패 시에도 실행되는 후처리 명령
image	잡용 Docker 이미지
rules	잡 포함 조건
needs	DAG 의존성 (스테이지 순서 건너뜀)
allow_failure	잡 실패 시에도 파이프라인 계속
retry	자동 재시도 횟수 (0-2)
timeout	최대 잡 실행 시간

Rules

```
deploy:
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
      when: manual
    - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'
      when: never
    - when: on_success
```

스테이지

스테이지 순서

```
stages:
  - lint
  - build
  - test
  - deploy
```

기본 스테이지

.pre	항상 가장 먼저 실행
build	기본 스테이지 1
test	기본 스테이지 2
deploy	기본 스테이지 3
.post	항상 마지막에 실행

needs를 이용한 DAG

```
test-api:
  stage: test
  needs: ["build-api"]    # skip waiting for full stage
test-web:
  stage: test
  needs: ["build-web"]    # runs as soon as build-web done
```

변수

변수 정의

```
variables:
  NODE_ENV: "production"
  DB_HOST: "postgres"
job:
  variables:
    NODE_ENV: "test"      # job-level override
```

사전 정의 변수

CI_COMMIT_SHA	전체 커밋 해시
CI_COMMIT_BRANCH	브랜치 이름
CI_COMMIT_TAG	태그 이름 (태그 파이프라인)
CI_PIPELINE_ID	고유 파이프라인 ID
CI_PROJECT_DIR	저장소 체크아웃 경로
CI_MERGE_REQUEST_IID	MR 번호 (MR 파이프라인 전용)
CI_REGISTRY_IMAGE	컨테이너 레지스트리 이미지 경로

보호됨 & 마스킹

Protected	보호된 브랜치/태그에서만 사용 가능
Masked	잡 로그에서 숨김
File	임시 파일에 기록; 경로가 변수에 담김

아티팩트

아티팩트 저장

```
build:
  script: npm run build
  artifacts:
    paths: [dist/]
    expire_in: 1 week
```

아티팩트 유형

paths	저장할 파일/디렉터리
exclude	제외할 패턴
expire_in	지정 기간 후 자동 삭제
reports:junit	MR 테스트 요약용 JUnit XML
reports:coverage_report	Cobertura 커버리지 시각화

JUnit 보고서

```
test:
  script: pytest --junitxml=report.xml
  artifacts:
    reports:
      junit: report.xml
```

캐시

의존성 캐싱

```
test:
  cache:
    key: ${CI_COMMIT_REF_SLUG}
    paths: [node_modules/]
  script: npm ci && npm test
```

캐시 vs 아티팩트

Cache	잡 속도 향상; 보장 안 됨; 동일 키 재사용
Artifacts	잡/스테이지 간 파일 전달; 보장됨

캐시 정책

pull-push	다운로드 + 업로드 (기본값)
pull	다운로드만 (소비자에게 빠름)
push	업로드만 (생산자용)

환경

환경 정의

```
deploy-staging:
  stage: deploy
  environment:
    name: staging
    url: https://staging.example.com
  script: ./deploy.sh staging
```

환경 기능

name	환경 이름 (UI에 표시)
url	배포된 앱 링크
on_stop	환경 중지 시 실행할 잡
auto_stop_in	지정 기간 후 자동 중지
action: stop	잡을 중지 동작으로 표시

리뷰 앱

```
review:
  environment:
    name: review/$CI_COMMIT_REF_SLUG
    url: https://$CI_COMMIT_REF_SLUG.example.com
    on_stop: stop-review
    auto_stop_in: 1 week
```

GitLab CI/CD 빠른 참조

Docker

이미지 빌드 & 푸시

```
build-image:
  image: docker:24
  services: [docker:24-dind]
  script:
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
$CI_REGISTRY
    - docker build -t $CI_REGISTRY_IMAGE:$CI_COMMIT_SHA .
    - docker push $CI_REGISTRY_IMAGE:$CI_COMMIT_SHA
```

서비스 (사이드카 컨테이너)

```
test:
  image: python:3.12
  services:
    - postgres:16
    - redis:7
  variables:
    POSTGRES_DB: testdb
    POSTGRES_PASSWORD: secret
```

Docker-in-Docker

`docker:24-dind`	DinD 서비스 이미지
`DOCKER_TLS_CERTDIR`	TLS 설정 시 '/certs' 또는 '' 로 설정
`DOCKER_HOST`	tcp://docker:2376 (TLS) 또는 :2375

일반 패턴

모노레포 (changes)

```
test-api:
  rules:
    - changes: [api/**/*]
test-web:
  rules:
    - changes: [web/**/*]
```

수동 배포 게이트

```
deploy-prod:
  stage: deploy
  when: manual
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
```

병렬 매트릭스

```
test:
  parallel:
    matrix:
      - PYTHON: ["3.10", "3.11", "3.12"]
        DB: ["postgres", "sqlite"]
  script: tox -e py${PYTHON}-${DB}
```