GitLab CI/CD クイックリファレンス

パイプライン、ジョブ、ステージ、変数、アーティファクト、環境

パイプラインの基本

パイプラインの仕組み

Pipeline	最上位コンテナ；コミット/トリガーごとに 1 つ
Stage	並列実行されるジョブのグループ
Job	ステージ内の単一タスク（スクリプト）
Runner	ジョブを実行するエージェント

パイプラインのトリガー

ブランチへのプッシュ	自動（デフォルト）
マージリクエスト	workflow:rules または only: merge_requests 経由
スケジュール	プロジェクト設定の CI/CD → Schedules
API	POST /projects/:id/trigger/pipeline
手動	CI/CD メニューの Run Pipeline ボタン

.gitlab-ci.yml

最小構成

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

グローバルキーワード

stages	ステージの順序を定義
default	すべてのジョブのデフォルト値
variables	グローバルな CI/CD 変数
workflow	パイプライン作成のタイミングを制御
include	外部 YAML ファイルをインポート

テンプレートのインクルード

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

ジョブ

ジョブの定義

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

ジョブキーワード

script	実行するシェルコマンド（必須）
before_script	メインスクリプト前のコマンド
after_script	後のコマンド（失敗時でも実行）
image	ジョブの Docker イメージ
rules	ジョブ含有の条件
needs	DAG 依存関係（ステージ順序をスキップ）
allow_failure	ジョブが失敗してもパイプラインを継続
retry	自動リトライ回数（0〜2）
timeout	最大ジョブ実行時間

Rules

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

ステージ

ステージの順序

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

デフォルトステージ

.pre	常に最初に実行
build	デフォルトステージ 1
test	デフォルトステージ 2
deploy	デフォルトステージ 3
.post	常に最後に実行

needs を使った DAG

```
test-api:
  stage: test
  needs: ["build-api"]    # ステージ全体を待たずに実行
test-web:
  stage: test
  needs: ["build-web"]    # build-web が完了次第実行
```

変数

変数の定義

```
variables:
  NODE_ENV: "production"
  DB_HOST: "postgres"
job:
  variables:
    NODE_ENV: "test"      # ジョブレベルの上書き
```

定義済み変数

CI_COMMIT_SHA	完全なコミットハッシュ
CI_COMMIT_BRANCH	ブランチ名
CI_COMMIT_TAG	タグ名（タグパイプラインの場合）
CI_PIPELINE_ID	一意なパイプライン ID
CI_PROJECT_DIR	リポジトリのチェックアウトパス
CI_MERGE_REQUEST_IID	MR 番号（MR パイプラインのみ）
CI_REGISTRY_IMAGE	コンテナレジストリイメージパス

Protected と Masked

Protected	保護されたブランチ/タグのみで利用可能
Masked	ジョブログで非表示
File	一時ファイルに書き込まれ、パスを変数に格納

アーティファクト

アーティファクトの保存

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

アーティファクトの種類

paths	保存するファイル/ディレクトリ
exclude	スキップするパターン
expire_in	一定期間後に自動削除
reports:junit	MR テストサマリー用 JUnit XML
reports:coverage_report	Cobertura カバレッジの可視化

JUnit レポート

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

キャッシュ

依存関係のキャッシュ

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

キャッシュ vs アーティファクト

Cache	ジョブを高速化；保証なし；同一キーを再利用
Artifacts	ジョブ/ステージ間でファイルを渡す；保証あり

キャッシュポリシー

pull-push	ダウンロード + アップロード（デフォルト）
pull	ダウンロードのみ（コンシューマーに高速）
push	アップロードのみ（プロデューサー用）

環境

環境の定義

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

環境の機能

name	環境名（UI に表示）
url	デプロイされたアプリへのリンク
on_stop	環境停止時に実行するジョブ
auto_stop_in	一定期間後に自動停止
action: stop	ジョブを停止アクションとしてマーク

レビューアプリ

```
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 クイックリファレンス

Docker

イメージのビルドとプッシュ

```
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
```

サービス（サイドカーコンテナ）

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

Docker-in-Docker

docker:24-dind	DinD サービスイメージ
DOCKER_TLS_CERTDIR	'/certs' または '' に設定して TLS を設定
DOCKER_HOST	tcp://docker:2376（TLS）または :2375

よくあるパターン

モノレポ（changes）

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

手動デプロイゲート

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

並列マトリクス

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