

# RIFERIMENTO RAPIDO GITLAB CI/CD

Pipeline, job, stage, variabili, artifact, ambienti

## Basi delle Pipeline

### Come Funzionano le Pipeline

**Pipeline** Contenitore principale; uno per commit/trigger

**Stage** Gruppo di job che girano in parallelo

**Job** Task singolo (script) all'interno di uno stage

**Runner** Agente che esegue i job

### Attivare le Pipeline

**Push su branch** Automatico (default)

**Merge request** Via workflow:rules o only: merge\_requests

**Pianificazione** CI/CD → Pianificazioni nelle impostazioni del

**API** POST /projects/:id/trigger/pipeline

**Manuale** Pulsante Esegui Pipeline nel menu CI/CD

## .gitlab-ci.yml

### Configurazione Minimale

```
stages: [build, test, deploy]
```

```
build-job:
```

```
  stage: build
```

```
  script: echo "Compilazione..."
```

### Keyword Globali

**stages** Definisce l'ordine degli stage

**default** Valori predefiniti per tutti i job

**variables** Variabili CI/CD globali

**workflow** Controlla quando le pipeline vengono create

**include** Importa file YAML esterni

### Include Template

```
include:
```

```
  - template: Auto-DevOps.gitlab-ci.yml
```

```
  - local: .ci/lint.yml
```

```
  - project: 'group/shared-ci'
```

```
    file: '/templates/deploy.yml'
```

## Job

### Definizione Job

```
test-unit:
```

```
  stage: test
```

```
  image: node:20
```

```
  script:
```

```
    - npm ci
```

```
    - npm test
```

### Keyword Job

**script** Comandi shell da eseguire (obbligatorio)

**before\_script** Comandi prima dello script principale

**after\_script** Comandi dopo (anche in caso di fallimento)

**image** Immagine Docker per il job

**rules** Condizioni per l'inclusione del job

**needs** Dipendenze DAG (salta l'ordine degli stage)

**allow\_failure** La pipeline continua anche se il job fallisce

**retry** Numero di tentativi automatici (0-2)

**timeout** Durata massima del job

### Regole

```
deploy:
```

```
  rules:
```

```
    - if: '$CI_COMMIT_BRANCH == "main"'
```

```
      when: manual
```

```
    - if: '$CI_PIPELINE_SOURCE == "merge_request_event"'
```

```
      when: never
```

```
    - when: on_success
```

## Stage

### Ordine degli Stage

```
stages:
```

```
  - lint
```

```
  - build
```

```
  - test
```

```
  - deploy
```

### Stage Predefiniti

**pre** Sempre eseguito per primo

**build** Stage predefinito 1

**test** Stage predefinito 2

**deploy** Stage predefinito 3

**post** Sempre eseguito per ultimo

### DAG con needs

```
test-api:
```

```
  stage: test
```

```
  needs: ["build-api"] # salta l'attesa dello stage completo
```

```
test-web:
```

```
  stage: test
```

```
  needs: ["build-web"] # esegue appena build-web è pronto
```

## Variabili

### Definire Variabili

```
variables:
```

```
  NODE_ENV: "production"
```

```
  DB_HOST: "postgres"
```

```
job:
```

```
  variables:
```

```
    NODE_ENV: "test" # override a livello job
```

### Variabili Predefinite

**CI\_COMMIT\_SHA** Hash completo del commit

**CI\_COMMIT\_BRANCH** Nome del branch

**CI\_COMMIT\_TAG** Nome del tag (se pipeline tag)

**CI\_PIPELINE\_ID** ID univoco della pipeline

**CI\_PROJECT\_DIR** Percorso checkout del repo

**CI\_MERGE\_REQUEST\_IID** Numero MR (solo pipeline MR)

**CI\_REGISTRY\_IMAGE** Percorso immagine container registry

### Protette e Mascherate

**Protetta** Disponibile solo su branch/tag protetti

**Mascherata** Nascosta nei log del job

**File** Scritta su file temporaneo; percorso nella variabile

## Artifact

### Salvare Artifact

```
build:
```

```
  script: npm run build
```

```
  artifacts:
```

```
    paths: [dist/]
```

```
    expire_in: 1 week
```

### Tipi di Artifact

**paths** File/directory da memorizzare

**exclude** Pattern da escludere

**expire\_in** Eliminazione automatica dopo la durata

**reports:junit** XML JUnit per il riepilogo test nella MR

**reports:coverage\_report** Visualizzazione copertura Cobertura

### Report JUnit

```
test:
```

```
  script: pytest --junitxml=report.xml
```

```
  artifacts:
```

```
    reports:
```

```
      junit: report.xml
```

## Cache

### Cache delle Dipendenze

```
test:
```

```
  cache:
```

```
    key: ${CI_COMMIT_REF_SLUG}
```

```
    paths: [node_modules/]
```

```
    script: npm ci && npm test
```

### Cache vs Artifact

**Cache** Velocizza i job; non garantita; riuso della stessa chiave

**Artifact** Passa file tra job/stage; garantita

### Policy Cache

**pull-push** Scarica + carica (default)

**pull** Solo scarica (più veloce per i consumatori)

**push** Solo carica (per i produttori)

## Ambienti

### Definire Ambienti

```
deploy-staging:
```

```
  stage: deploy
```

```
  environment:
```

```
    name: staging
```

```
    url: https://staging.example.com
```

```
    script: ./deploy.sh staging
```

### Funzionalità Ambiente

**name** Nome ambiente (mostrato nell'interfaccia)

**url** Link all'app deployata

**on\_stop** Job da eseguire quando l'ambiente viene fermato

**auto\_stop\_in** Fermata automatica dopo la durata

**action: stop** Marca il job come azione di stop

### Review App

```
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
```

## Docker

### Build e Push Immagine

```
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
```

### Servizi (Container Sidecar)

```
test:
```

```
  image: python:3.12
```

```
  services:
```

```
    - postgres:16
```

```
    - redis:7
```

```
  variables:
```

```
    POSTGRES_DB: testdb
```

```
    POSTGRES_PASSWORD: secret
```

### Docker-in-Docker

**docker:24-dind** Immagine servizio DinD

**DOCKER\_TLS\_CERTDIR** Impostare su '/certs' o '' per config TLS

**DOCKER\_HOST** tcp://docker:2376 (TLS) o :2375

### Pattern Comuni

### Monorepo (changes)

```
test-api:
```

```
  rules:
```

```
    - changes: [api/**/*]
```

```
test-web:
```

```
  rules:
```

```
    - changes: [web/**/*]
```

### Gate Deploy Manuale

```
deploy-prod:
```

```
  stage: deploy
```

```
  when: manual
```

```
  rules:
```

```
    - if: '$CI_COMMIT_BRANCH == "main"'
```

### Matrix Parallela

```
test:
```

```
  parallel:
```

```
    matrix:
```

```
      - PYTHON: ["3.10", "3.11", "3.12"]
```

```
        DB: ["postgres", "sqlite"]
```

```
    script: tox -e py${PYTHON}-${DB}
```