

GitLab CI/CD Quick Reference

Pipelines, jobs, stages, variables, artifacts, environments

Pipeline Basics

How Pipelines Work

Pipeline	Top-level container; one per commit/trigger
Stage	Group of jobs that run in parallel
Job	Single task (script) within a stage
Runner	Agent that executes jobs

Triggering Pipelines

Push to branch	Automatic (default)
Merge request	Via workflow:rules or only: merge_requests
Schedule	CI/CD → Schedules in project settings
API	POST /projects/:id/trigger/pipeline
Manual	Run Pipeline button in CI/CD menu

.gitlab-ci.yml

Minimal Config

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

Global Keywords

stages	Define stage order
default	Default values for all jobs
variables	Global CI/CD variables
workflow	Control when pipelines are created
include	Import external YAML files

Include Templates

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

Jobs

Job Definition

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

Job Keywords

script	Shell commands to run (required)
before_script	Commands before main script
after_script	Commands after (even on failure)
image	Docker image for the job
rules	Conditions for job inclusion
needs	DAG dependencies (skip stage order)
allow_failure	Pipeline continues if job fails
retry	Auto-retry count (0-2)
timeout	Max job duration

Rules

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

Stages

Stage Order

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

Default Stages

.pre	Always runs first
build	Default stage 1
test	Default stage 2
deploy	Default stage 3
.post	Always runs last

DAG with needs

```
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
```

Variables

Defining Variables

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

Predefined Variables

CI_COMMIT_SHA	Full commit hash
CI_COMMIT_BRANCH	Branch name
CI_COMMIT_TAG	Tag name (if tag pipeline)
CI_PIPELINE_ID	Unique pipeline ID
CI_PROJECT_DIR	Repo checkout path
CI_MERGE_REQUEST_IID	MR number (MR pipelines only)
CI_REGISTRY_IMAGE	Container registry image path

Protected & Masked

Protected	Only available on protected branches/tags
Masked	Hidden in job logs
File	Written to a temp file; path in variable

Artifacts

Saving Artifacts

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

Artifact Types

paths	Files/dirs to store
exclude	Patterns to skip
expire_in	Auto-delete after duration
reports:junit	JUnit XML for MR test summary
reports:coverage_report	Cobertura coverage visualization

JUnit Report

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

Cache

Caching Dependencies

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

Cache vs Artifacts

Cache	Speed up jobs; not guaranteed; same key reuse
Artifacts	Pass files between jobs/stages; guaranteed

Cache Policies

pull-push	Download + upload (default)
pull	Download only (faster for consumers)
push	Upload only (for producers)

Environments

Defining Environments

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

Environment Features

name	Environment name (shown in UI)
url	Link to deployed app
on_stop	Job to run when environment is stopped
auto_stop_in	Auto-stop after duration
action: stop	Marks job as the stop action

Review Apps

```
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 & Push Image

```
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
```

GitLab CI/CD Quick Reference

Services (Sidecar Containers)

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

Docker-in-Docker

docker:24-dind	DinD service image
DOCKER_TLS_CERTDIR	Set to '/certs' or '' for TLS config
DOCKER_HOST	tcp://docker:2376 (TLS) or :2375

Common Patterns

Monorepo (changes)

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

Manual Deploy Gate

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

Parallel Matrix

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