# GitLab CI/CD Pipelines in YAML

## CONCEPTS

Everything lives in .gitlab-ci.yml at the repo root; a push or merge request creates a pipeline of stages, each holding parallel jobs.

**Pipeline** one per push, MR or schedule

- **Stage** runs in order; next waits for all jobs
  - **Job** script run by a runner in a clean checkout

Jobs start clean: pass files with artifacts, speed up installs with cache (best effort only).

## JOBS

`<job>:`
`  script:`
`    - <cmd>`
Minimal job

`stage: <name>`
Put the job in a stage

`image: <name>:<tag>`
Container image for the job

`before_script: [<cmd>]`
Run before script

`after_script: [<cmd>]`
Run after, even on failure

`services: [postgres:16]`
Sidecar containers (databases)

## JOBS (cont.)

`tags: [<runner-tag>]`
Choose runners by tag

`timeout: 30m`
Fail the job after this long

`retry: 2`
Retry failed jobs

`allow_failure: true`
Let the job fail without failing the pipeline

`when: manual`
Wait for a click to run

## STAGES AND ORDER

`stages: [build, test, deploy]`
Set stage order

`needs: [<job>]`
Start once these jobs finish

`dependencies: [<job>]`
Fetch artifacts only from these jobs

## RULES

`rules: - if: $CI_COMMIT_BRANCH == "main"`
Run when the expression is true

`rules: - if: $CI_PIPELINE_SOURCE == "merge_request_event"`
Run in merge request pipelines

## RULES (cont.)

`rules: - changes: ["src/**"]`
Run when these files changed

`rules: - when: never`
Fallback: skip the job

`rules: - if: <expr>`
`    when: manual`
Manual only on a condition

`workflow:`
`  rules: [...]`
Decide whether a pipeline exists

## ARTIFACTS AND CACHE

`artifacts:`
`  paths: [<path>]`
Keep files after the job

`artifacts: expire_in: 1 week`
Expire artifacts

`artifacts: when: always`
Upload even on failure

`artifacts: reports: junit: <file>`
Show test results in the MR

`cache:`
`  paths: [<dir>]`
Reuse files between runs

`cache: key: $CI_COMMIT_REF_SLUG`
One cache per branch

## ARTIFACTS AND CACHE (cont.)

`cache: key: {files: [<lockfile>]}`
Key on lockfile contents

## REUSE AND INCLUDE

`include: - local: <path>`
Include a file from this repo

`include: - project: <group/proj>`
`    file: <path>`
Include from another project

`include: - template: <name>.gitlab-ci.yml`
Use a GitLab template

`<job>:`
`  extends: .<template>`
Inherit from a hidden job

`.<name>:`
Hidden job: a template, never runs

`default: {image: <img>}`
Defaults for all jobs

`parallel:`
`  matrix:`
`    - <VAR>: [a, b]`
One job per combination

## VARIABLES AND ENVIRONMENTS

`variables:`
`  <NAME>: <value>`
Pipeline or job variable

## VARIABLES AND ENVIRONMENTS (cont.)

`$CI_COMMIT_BRANCH`
Branch name

`$CI_COMMIT_SHA`
Commit hash

`$CI_COMMIT_REF_SLUG`
URL-safe ref name

`$CI_PIPELINE_SOURCE`
Why the pipeline ran (push, merge_request_event, ...)

`$CI_DEFAULT_BRANCH`
The project's default branch

`$CI_PROJECT_DIR`
Checkout directory

`environment: {name: <env>, url: <url>}`
Track a deployment

Secrets go in Settings > CI/CD > Variables (masked, protected), not in the YAML.

## RECIPES

**Deploy on main with approval**

```
deploy:
  stage: deploy
  environment: production
  script: ./deploy.sh
  rules:
    - if: $CI_COMMIT_BRANCH
== $CI_DEFAULT_BRANCH
      when: manual
```