

Referência Rápida GitLab CI/CD

Pipelines, jobs, stages, variáveis, artifacts, ambientes

Fundamentos de Pipeline

Como Pipelines Funcionam

Pipeline	Container de nível superior; um por commit/gatilho
Stage	Grupo de jobs que rodam em paralelo
Job	Tarefa única (script) dentro de um stage
Runner	Agente que executa jobs

Acionando Pipelines

Push to branch	Automático (padrão)
Merge request	Via workflow:rules ou only: merge_requests
Schedule	CI/CD → Schedules nas configurações do projeto
API	POST /projects/:id/trigger/pipeline
Manual	Botão Run Pipeline no menu CI/CD

.gitlab-ci.yml

Configuração Mínima

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

Palavras-chave Globais

stages	Definir ordem dos stages
default	Valores padrão para todos os jobs
variables	Variáveis CI/CD globais
workflow	Controlar quando pipelines são criados
include	Importar arquivos YAML externos

Incluir Templates

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

Jobs

Definição de Job

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

Palavras-chave de Job

script	Comandos shell a executar (obrigatório)
before_script	Comandos antes do script principal
after_script	Comandos após (mesmo em falha)
image	Imagem Docker para o job
rules	Condições para inclusão do job
needs	Dependências DAG (pular ordem de stage)
allow_failure	Pipeline continua se job falhar
retry	Contagem de retry automático (0-2)
timeout	Duração máxima do job

Rules

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

Stages

Ordem dos Stages

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

Stages Padrão

.pre	Sempre executa primeiro
build	Stage padrão 1
test	Stage padrão 2
deploy	Stage padrão 3
.post	Sempre executa por último

DAG com 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
```

Variáveis

Definindo Variáveis

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

Variáveis Predefinidas

CI_COMMIT_SHA	Hash completo do commit
CI_COMMIT_BRANCH	Nome da branch
CI_COMMIT_TAG	Nome da tag (se pipeline de tag)
CI_PIPELINE_ID	ID único do pipeline
CI_PROJECT_DIR	Caminho de checkout do repositório
CI_MERGE_REQUEST_IID	Número do MR (apenas pipelines de MR)
CI_REGISTRY_IMAGE	Caminho da imagem no registro de containers

Protegidas e Mascaradas

Protected	Disponível apenas em branches/tags protegidas
Masked	Ocultas nos logs do job
File	Escrita em arquivo temporário; caminho na variável

Artifacts

Salvando Artifacts

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

Tipos de Artifact

paths	Arquivos/diretórios para armazenar
exclude	Padrões a ignorar
expire_in	Auto-excluir após duração
reports:junit	XML JUnit para resumo de testes no MR
reports:coverage_report	Visualização de cobertura Cobertura

Relatório JUnit

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

Cache

Cacheando Dependências

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

Cache vs Artifacts

Cache	Acelerar jobs; não garantido; reutilização por chave
Artifacts	Passar arquivos entre jobs/stages; garantido

Políticas de Cache

pull-push	Baixar + enviar (padrão)
pull	Apenas baixar (mais rápido para consumidores)
push	Apenas enviar (para produtores)

Ambientes

Definindo Ambientes

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

Recursos de Ambiente

name	Nome do ambiente (exibido na UI)
url	Link para a app implantada
on_stop	Job a executar quando o ambiente é parado
auto_stop_in	Parar automaticamente após duração
action: stop	Marca job como ação de parada

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

Construir e Publicar Imagem

```
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
```

Referência Rápida GitLab CI/CD

Services (Containers Sidecar)

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

Docker-in-Docker

docker:24-dind	Imagem do serviço DinD
DOCKER_TLS_CERTDIR	Definir como '/certs' ou '' para configuração TLS
DOCKER_HOST	tcp://docker:2376 (TLS) ou :2375

Padrões Comuns

Monorepo (changes)

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

Portão Manual de Deploy

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

Matrix Paralela

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