

# GITLAB CI/CD 快速参考

流水线、作业、阶段、变量、产物、环境

## 流水线基础

### 流水线工作原理

**Pipeline** 顶层容器；每次 commit/触发一个  
**Stage** 并行运行的一组作业  
**Job** 阶段内的单个任务（脚本）  
**Runner** 执行作业的代理

### 触发流水线

**Push to branch** 自动触发（默认）  
**Merge request** 通过 workflow:rules 或 only: merge\_requests  
**Schedule** 项目设置 CI/CD → Schedules  
**API** POST /projects/:id/trigger/pipeline  
**Manual** CI/CD 菜单中的 Run Pipeline 按钮

## gitlab-ci.yml

### 最简配置

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

### 全局关键字

**stages** 定义阶段顺序  
**default** 所有作业的默认值  
**variables** 全局 CI/CD 变量  
**workflow** 控制何时创建流水线  
**include** 导入外部 YAML 文件

### 引入模板

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

## 作业

### 作业定义

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

### 作业关键字

**script** 要执行的 shell 命令（必需）  
**before\_script** 主脚本前执行的命令  
**after\_script** 主脚本后执行（即使失败）  
**image** 作业使用的 Docker 镜像  
**rules** 作业包含的条件  
**needs** DAG 依赖（跳过阶段顺序）  
**allow\_failure** 作业失败时流水线继续  
**retry** 自动重试次数（0-2）  
**timeout** 最大作业时长

### Rules

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

## 阶段

### 阶段顺序

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

### 默认阶段

**pre** 始终最先运行  
**build** 默认阶段 1  
**test** 默认阶段 2  
**deploy** 默认阶段 3  
**post** 始终最后运行

### DAG 与 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
```

## 变量

### 定义变量

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

### 预定义变量

**CI\_COMMIT\_SHA** 完整 commit hash  
**CI\_COMMIT\_BRANCH** 分支名称  
**CI\_COMMIT\_TAG** Tag 名称 (tag 流水线)  
**CI\_PIPELINE\_ID** 唯一流水线 ID  
**CI\_PROJECT\_DIR** 仓库检出路径  
**CI\_MERGE\_REQUEST\_ID** MR 编号 (仅 MR 流水线)  
**CI\_REGISTRY\_IMAGE** 容器镜像仓库路径

### 保护与脱敏

**Protected** 仅在受保护的分支/tag 上可用  
**Masked** 在作业日志中隐藏

**File** 写入临时文件；变量为文件路径

## 产物

### 保存产物

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

### 产物类型

**paths** 要存储的文件/目录  
**exclude** 要跳过的模式  
**expire\_in** 持续时间长后自动删除  
**reports:junit** JUnit XML, 用于 MR 测试摘要  
**reports:coverage\_report** Cobertura 覆盖率可视化

### JUnit 报告

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

## 缓存

### 缓存依赖

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

### 缓存 vs 产物

**Cache** 加速作业；不保证命中；同 key 复用  
**Artifacts** 在作业/阶段间传递文件；有保证

### 缓存策略

**pull-push** 下载 + 上传（默认）  
**pull** 仅下载（消费者更快）  
**push** 仅上传（生产者）

## 环境

### 定义环境

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

### 环境特性

**name** 环境名称（在 UI 中显示）  
**url** 已部署应用的链接  
**on\_stop** 停止环境时执行的作业  
**auto\_stop\_in** 持续时间长后自动停止  
**action: stop** 将作业标记为停止动作

### 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

### 构建并推送镜像

```
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
```

### 服务 (Sidecar 容器)

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

### Docker-in-Docker

**docker:24-dind** DinD 服务镜像  
**DOCKER\_TLS\_CERTDIR** 设为 '/certs' 或 " 配置 TLS  
**DOCKER\_HOST** tcp://docker:2376 (TLS) 或 :2375

### 常用模式

#### Monorepo (changes)

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

#### 手动部署门控

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

#### 并行矩阵

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