# Рекомендуемый порядок расположения блоков внутри job в .gitlab-ci.yml?

В GitLab CI/CD **есть рекомендованный порядок**, который улучшает **читаемость и поддержку pipeline**.

---

## □□ Стандартный порядок блоков внутри `job`

```
job_name:
  extends: .some_template  # 1□ Наследование (если есть)
  stage: build             # 2□ Определение этапа
  tags:                    # 3□ Теги для раннера (если нужны)
    - docker
  needs:                   # 4□ Зависимости от других job'ов
    - previous_job
  variables:               # 5□ Локальные переменные для job
    SOME_VAR: "value"
  before_script:           # 6□ Подготовка окружения перед `script`
    - echo "Setting up environment..."
  script:                  # 7□ Основной код
    - echo "Executing job..."
  after_script:            # 8□ Завершающие действия (если нужны)
    - echo "Cleanup after job"
  artifacts:               # 9□ Артефакты (если нужны)
    paths:
      - my_output/
    expire_in: 1 hour
```

```
rules:                # □□ Условия выполнения job
  - if: '$CI_COMMIT_BRANCH == "main"'
allow_failure: false      # 1□1□ Позволять ли падение job
retry: 2                  # 1□2□ Число повторных попыток
timeout: 30m              # 1□3□ Лимит времени выполнения
```

---

# □□ Разбор структуры

| Секция | Описание |
|---|---|
| `extends:` | Наследование от шаблонного `job` (если есть). |
| `stage:` | Назначение `job` к определенному этапу. |
| `tags:` | Указывает, на каких раннерах будет выполняться `job`. |
| `needs:` | Указывает зависимости от других `job` (если надо запустить `job` только после выполнения других). |
| `variables:` | Определяет переменные, видимые только внутри `job`. |
| `before_script:` | Выполняется **один раз перед** `script:` (установка окружения, логирование, авторизация и т. д.). |
| `script:` | Основной код `job`, выполняется **после** `before_script:`. |
| `after_script:` | Выполняется **после** `script:`, даже если `script:` упал. Подходит для логов, очистки файлов. |
| `artifacts:` | Определяет файлы, которые сохраняются после завершения `job`. |
| `rules:` | Условия выполнения `job` (например, только в `main`, только при изменениях в `src/`). |
| `allow_failure:` | Разрешает `job` завершаться с ошибкой (`true` → ошибки не остановят pipeline). |
| `retry:` | Число повторных попыток при падении `job`. |
| `timeout:` | Лимит времени выполнения `job` (например, `30m`). |

---

# □□ Пример 1: Минимальная конфигурация `job`

Если у вас простой `job`, можно оставить только **ключевые секции**:

```
build_job:
  stage: build
  script:
    - echo "Building the project..."
```

**Минимально необходимый `job`**: `stage:` + `script:`.

---

# Пример 2: Расширенная конфигурация `job`

Если `job` сложный, используем **все ключевые блоки**:

```
deploy_job:
  extends: .deploy_template
  stage: deploy
  tags:
    - production-runner
  needs:
    - build_job
  variables:
    DEPLOY_ENV: "production"
  before_script:
    - echo "Подготовка к деплою..."
  script:
    - echo "Выполняем деплой..."
  after_script:
    - echo "Очистка после деплоя..."
  artifacts:
    paths:
      - deploy_logs/
    expire_in: 1 day
  rules:
    - if: '$CI_COMMIT_BRANCH == "main"'
  allow_failure: false
```

```
retry: 2
timeout: 20m
```

 Этот `job` **должен выполниться ТОЛЬКО в** `main`.
**Если** `job` **упадет, он попробует перезапуститься дважды** (`retry: 2`).
[ **`before_script:` подготавливает окружение, а** `after_script:` **выполняет очистку.**

---

# Итог

# Оптимальный порядок блоков `job`:

1[ `extends:` (если используется)
2[ `stage:`
3[ `tags:` (если есть)
4[ `needs:` (если job зависит от других job'ов)
5[ `variables:` (локальные переменные)
6[ `before_script:` (подготовка окружения)
7[ `script:` (основная логика job)
8[ `after_script:` (очистка, логи)
9[ `artifacts:` (если нужны)
[ `rules:` (условия выполнения)
1 1 `allow_failure:` (можно ли игнорировать ошибки)
1 2 `retry:` (количество повторных попыток)
1 3 `timeout:` (ограничение времени выполнения)

---

Revision #1
Created 31 January 2025 11:20:18 by Admin
Updated 31 January 2025 11:23:59 by Admin