a w sekcji workflow

Niestety, w sekcji `workflow` nie możesz bezpośrednio sprawdzić, czy tag znajduje się na konkretnej gałęzi, ponieważ `workflow:rules` nie obsługuje wykonywania skryptów Git. Sekcja `workflow` jest oceniana przed utworzeniem pipeline'a i ma dostęp tylko do zmiennych CI/CD, nie do komend Git.[1] [2]

Ograniczenia workflow:rules

W `workflow:rules` możesz używać tylko warunków opartych na zmiennych środowiskowych jak `$CI_COMMIT_TAG`, `$CI_COMMIT_BRANCH`, czy `$CI_PIPELINE_SOURCE`. Git nie tworzy bezpośredniej relacji między tagiem a branchem - są to osobne referencje wskazujące na commit. Dlatego nie ma zmiennej CI/CD, która bezpośrednio wskazywałaby "branch, na którym znajduje się tag".[2] [3] [1]

Rozwiązania obejściowe

Masz trzy możliwe podejścia:

1. Konwencja nazewnictwa tagów

Możesz użyć konwencji nazewnictwa tagów, aby odróżnić tagi produkcyjne (z mastera) od innych:[2]

```
workflow:
  rules:
    - if: '$CI_COMMIT_TAG =~ /^v\d+\.\d+\.\d+$/'
      # Uruchom tylko dla tagów typu v1.2.3 (semantic versioning)
    - if: $CI_COMMIT_BRANCH == "master"
    - when: never
```

To wymaga dyscypliny w tworzeniu tagów - tylko tagi na masterze używają pełnego semantic versioning, a tagi na innych branchach używają innych konwencji.[2]

2. Weryfikacja w before_script pierwszego joba

Wykonaj sprawdzenie w pierwszym jobie pipeline'a, który natychmiast zakończy się błędem, jeśli tag nie jest na masterze:[2]

```
workflow:
  rules:
    - if: $CI_COMMIT_TAG
    - if: $CI_COMMIT_BRANCH == "master"
```

```
    - when: never

verify-tag-location:
  stage: .pre
  rules:
    - if: $CI_COMMIT_TAG
  before_script:
    - git fetch origin master:master
  script:
    - |
      if ! git merge-base --is-ancestor $CI_COMMIT_SHA origin/master; then
        echo "❌ Tag musi być na branchu master"
        exit 1
      fi
    - echo "✅ Tag jest na master - pipeline może kontynuować"

build-job:
  stage: build
  needs: ["verify-tag-location"]
  script:
    - echo "Building..."
```

Stage `.pre` jest specjalnym stage'm wykonywanym przed wszystkimi innymi. Jeśli weryfikacja się nie powiedzie, cały pipeline zostanie anulowany.[4]

3. Ochrona brancha i polityka tagów

Najlepszym rozwiązaniem jest skonfigurowanie ochrony brancha master w ustawieniach GitLab, aby tylko określone osoby mogły tworzyć tagi na masterze. Następnie w `workflow` możesz ufać, że wszystkie tagi są tworzone poprawnie:[5] [2]

```
workflow:
  rules:
    - if: $CI_COMMIT_TAG  # Zaufaj, że tagi są tworzone tylko na master
    - if: $CI_COMMIT_BRANCH == "master"
    - when: never
```

To organizacyjne rozwiązanie jest najprostsze i najbezpieczniejsze, ponieważ wymusza poprawne praktyki na poziomie uprawnień GitLab.[2]

⁂

1. https://docs.gitlab.com/ci/yaml/workflow/
2. https://stackoverflow.com/questions/42796018/how-to-run-a-gitlab-ci-yml-job-only-on-a-tagged-branch
3. https://gitlab.cn/docs/en/ee/ci/yaml/workflow.html
4. https://docs.gitlab.com/ci/jobs/job_rules/
5. https://docs.gitlab.com/user/project/repository/tags/
6. https://www.reddit.com/r/gitlab/comments/1dtmfi4/new_branch_run_job_if_changes_in_path/
7. https://microfluidics.utoronto.ca/gitlab/help/ci/yaml/workflow.md

8. https://gitlab.com/gitlab-org/gitlab/-/issues/15170
9. https://docs.gitlab.com/user/project/repository/branches/
10. https://meleu.dev/notes/gitlab-ci-rules/