"GitLab CI: pipeline en rojo por cuenta bloqueada (failure_reason: user_blocked)"

- **Fecha:** 2026-06-18
- **Autor:** Abdelkarim Mateos
- **Estado:** vigente
- **Ámbito:** GitLab CI/CD, offboarding de desarrolladores

Resumen

Un pipeline de GitLab CI/CD puede aparecer como fallido de forma **engañosa** cuando la cuenta del usuario que lo disparó (normalmente el autor del merge request) se **bloquea o deshabilita mientras el pipeline está en ejecución**. GitLab aborta el job en vuelo y le asigna `failure_reason: user_blocked`. No es un fallo del código ni de la infraestructura: es un efecto colateral del offboarding.

Síntoma

- Un job de CI aparece como `failed`.
- El trace del job **se corta a mitad de ejecución**, sin el resumen final de la herramienta (no aparece `Tests: X failed`), sin excepción ni assert.
- El **mismo commit pasa en verde en local**.
- Se confunde fácilmente con: OOM o contención del runner, test flaky, race condition, o un bug real.

Causa

El offboarding de un desarrollador incluye bloquear su cuenta de GitLab. Si ese usuario tenía un pipeline corriendo (típicamente el de su propio MR), al bloquear la cuenta GitLab mata ese pipeline. El job queda en `failed` con `failure_reason = user_blocked`. El tiempo transcurrido del job puede ser alto (corrió varios minutos hasta el bloqueo), lo que despista aún más.

Diagnóstico: el dato que clasifica (no teorizar antes)

Antes de tocar código, tests o infraestructura, leer el `failure_reason` del job:

```
glab api "projects/<grupo>%2F<proyecto>/jobs/<job_id>" \

  | python3 -c "import sys,json;print(json.load(sys.stdin)['failure_reason'])"
```

Valores típicos y su significado:

- `script_failure` — un comando o test falló de verdad. Hay que leer el log.
- `runner_system_failure` / `stuck_or_timeout_failure` — problema de runner o infraestructura.
- `job_execution_timeout` — el job superó su timeout.
- `user_blocked` — la cuenta dueña del pipeline fue bloqueada. **Ni código ni infraestructura.**

Confirmar el estado de la cuenta (`active` o `blocked`):

```
glab api "users?search=<usuario>" \

  | python3 -c "import sys,json;[print(u['username'], u['state']) for u in
json.load(sys.stdin)]"
```

Solución

1. Re-lanzar el pipeline desde una cuenta **activa** (no la bloqueada). Para un MR, sin necesidad de push ni de commit basura:

   ```
   glab api --method POST "projects/<grupo>%2F<proyecto>/merge_requests/<iid>/pipelines"
   ```

   Crea un pipeline nuevo de tipo `merge_request_event` bajo la identidad del token usado.
2. Un `retry` del pipeline original puede heredar la identidad bloqueada y volver a fallar. Es preferible **crear un pipeline nuevo** desde cuenta activa, o rebasar/pushear la rama desde cuenta activa.
3. Los MRs de un usuario bloqueado **siguen siendo mergeables por un maintainer o admin**; solo hay que revalidar el CI bajo cuenta activa antes de integrar.

Regla derivada

- Clasificar **siempre** un fallo de CI por su `failure_reason` antes de tocar código, tests o infraestructura. El tipo de error discrimina la causa desde el primer vistazo y evita horas de diagnóstico equivocado.
- Trace truncado sin assert ni excepción **+ verde en local** = sospechar de un **aborto externo** (usuario bloqueado, pipeline cancelado, runner caído), no de un bug.
- Checklist de offboarding: al bloquear la cuenta de un desarrollador, re-disparar sus pipelines y MRs en vuelo desde una cuenta activa para no dejar el CI falsamente en rojo, que bloquearía merges legítimos por la regla `ci_must_pass`.

Revision #1
Created 2026-06-18 15:39:59 UTC by Abkrim
Updated 2026-06-18 15:39:59 UTC by Abkrim