

Q



DevOps (</blog/tags.html#DevOps>)

May 20, 2021 · 5 min read · [Leave a comment](#)



(/company/team/#dhershkovitch)

[Dov Hershkovitch \(/company/team/#dhershkovitch\)](/company/team/#dhershkovitch)



In 13.12 we fixed a bug (<https://gitlab.com/gitlab-org/gitlab/-/issues/31264>) that might affect the existing behavior of your pipeline. We explain why we had to fix the bug, the possible impact of this change on your pipeline, and the proposed workaround if you would like to revert this behavior.

Hi there! What brings you to the site today?

In GitLab CI/CD you can easily configure a job to require manual intervention before it runs. The job gets added to the pipeline, but doesn't run until you click the **play** button on it.


Let's look at a two-job pipeline:

```
stages:
  - stage1
  - stage2

job1:
  stage: stage1
  script:
    - echo "this is an automatic job"

manual_job:
  stage: stage2
  script:
    - echo "This is a manual job which doesn't start automatically, and the pipeline can complete without it starting."
  when: manual # This setting turns a job into a manual one
```

This is how it looks when we look at the pipeline graph:

 **passed** Pipeline #301003883 triggered 17 seconds ago by  **Dov HersHKovitch** 🙋🙋

## Update .gitlab-ci.yml file

🕒 2 jobs for **master** (queued for 2 seconds)

🚩 **latest**

🔑 **673abee4** 📁

🔗 No related merge requests found.

**Pipeline** Needs Jobs **2** Tests **0**

**Stage1**

**Stage2**



job1



manual\_job



Notice that the manual job gets skipped, and the pipeline completes successfully even though the manual job did not get triggered. This happens because manual jobs are considered optional, and do not need to run.

Hi there! What brings you to the site today?

1

Internally, manual jobs have `allow_failure` set to true by default, which means that these skipped manual jobs do not cause a pipeline failure. The YAML code below demonstrates how to write the manual job, which results in the same behavior. The job doesn't automatically start, is skipped, and the pipeline passes.

```
manual_job:
  stage: stage2
  script:
    - echo "This is a manual job which doesn't start automatically, and the pipeline can complete without it starting."
  when: manual
  allow_failure: true # this line is redundant since manual job has this setting by default
```

You can set `allow_failure` to true for any job, including both manual and automatic jobs, and then the pipeline does not care if the job runs successfully or not.

## How to expand the configuration with `needs` (DAG)

Last year we introduced the `needs` keyword which lets you create a Directed Acyclic Graphs (DAG) to speed up your pipeline (<https://docs.gitlab.com/ee/ci/yaml/#needs>). The `needs` keyword creates a dependency between two jobs regardless of their stage.

Let's look at this example:

```
stages:
  - stage1
  ....
  - stage10

job1: # this is the first job that runs in the pipeline
  stage: stage1
  script:
    - echo "exit 0"
  ....

job10:
  needs: # Defined a "needs" relationship with job1
    - job1
  stage: stage10
  script:
    - echo "This job runs as soon as job1 completes, even though this job is in stage10."
```

The `needs` keyword creates a dependency between the two jobs, so `job10` runs as soon as `job1` **finishes running** successfully, regardless of the stage ordering.

So what happens if a job `needs` a manual job, that doesn't start running automatically?

Let's look at the following example:



Hi there! What brings you to the site today?

1

```
stages:
  - build
  - test
  - deploy

build:
  stage: build
  script: exit 0

test:
  stage: test
  when: manual
  script: exit 0

deploy:
  stage: deploy
  script: echo "when should this job run?"
  needs:
    - test
```

Before 13.12, this type of configuration would cause the pipeline to get stuck. The `deploy` job can only start when the `test` job completes, but the `test` job does not start automatically. The rest of the pipeline stops and waits for someone to run the manual `test` job.

running Pipeline #2169 triggered 13 seconds ago by Administrator

Cancel running

Delete

## Update .gitlab-ci.yml

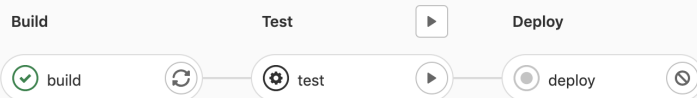
🕒 3 jobs for `master` (queued for 4 seconds)

📄 latest

🔗 75b69e19

🔍 No related merge requests found.

Pipeline Needs Jobs 3 Tests 0



This behavior is even worse with larger pipelines:

Hi there! What brings you to the site today?

1

running Pipeline #763 triggered 18 seconds ago by Administrator
 Cancel running Delete

### Update .gitlab-ci.yml

🕒 4 jobs for **master** (queued for 5 seconds)

🏷️ latest

🔑 bc4d0559

🔍 No related merge requests found.

Pipeline Needs Jobs 4 Tests 0

Build

Test

Post\_test

Deploy

✓ build

🔄

⚙️ test

▶️

⬛ post\_test

🚫

⬛ deploy

🚫

The example above shows there is a needs relationship between `post_test` job and the `test` job (which is a manual job) as you can see the pipeline is stuck in a running state and any subsequent jobs will not run.

This was not the behavior most users expected, so we improved it in 13.12. Now, if there is a `needs` relationship pointing to a manual job, the pipeline doesn't stop by default anymore. The manual job is considered optional by default in all cases now. Any jobs that have a `needs` relationship to manual jobs are now also considered optional and skipped if the manual job isn't triggered. If you start the manual job, the jobs that need it can start after it completes.

Note that if you start the manual job before a later job that has it in a `needs` configuration, the later job will still wait for the manual job to finishes running.

## What if I don't want this new behavior?

One of the reasons we selected this solution is that you can quickly revert this change. If you made use of this inadvertent behavior and configured your pipelines to use it to block on manual jobs, it's easy to return to that previous behavior. All you have to do is override the default `allow_failure` in the manual job with `allow_failure: false`. This way the manual job is no longer optional, and the pipeline status will be marked as blocked and wait for you to run the job manually.

```
stages:
  - build
  - test
  - deploy

build:
  stage: build
  script: exit 0

test:
  stage: test
  when: manual
  allow_failure: false # Set to false to return to the previous behavior.
  script: exit 0

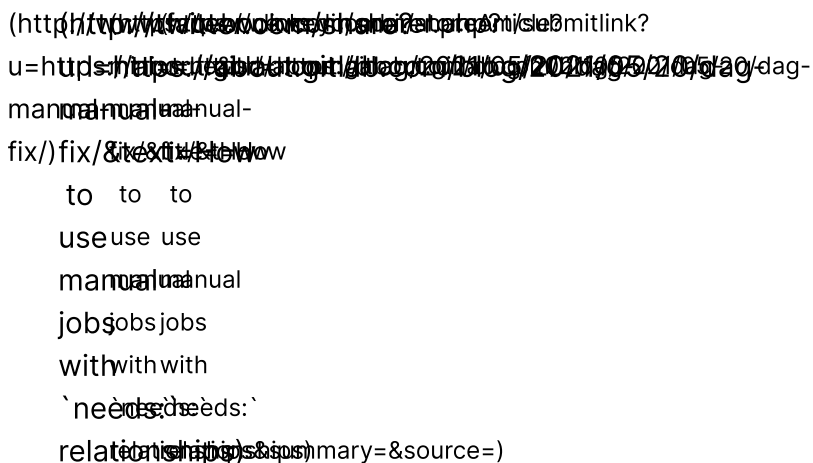
deploy:
  stage: deploy
  script: exit 0
  needs:
    - test
```

Share any thoughts, comments, or questions, by opening an issue in GitLab and mentioning me. Hi there! What brings you to the site today?

 Click to tweet


(t=%E2%80%9CText+to+tweet%E2%80%9D+%E2%80%93+40dov0211&url=https://about.gitlab.com/blog/2021/05/20/dag-anual-fix/)

[Learn more \(/resources/guide-to-the-cloud/\)](/resources/guide-to-the-cloud/)




[DevOps \(/blog/tags.html#DevOps\)](/blog/tags.html#DevOps)


 ENGINEERING

 Matej Latin

 ENGINEERING

 David O'Regan

 ENGINE



Dov

(/blog/2021/02

Hi there! What brings you to the site today?

# Try all GitLab features - free for 30 days

GitLab is more than just source code management or CI/CD  
(<https://about.gitlab.com/topics/ci-cd/>). It is a full software development lifecycle & DevOps tool in a single application.

Try GitLab Free (/free-trial/)

6 Comments

GitLab

Disqus' Privacy Policy

Login

Favorite 1

Tweet

Share

Sort by Best

Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Michael • 3 months ago

IMO the funamental flaw here is the obfuscation around what it means to be "optional" vs "required". As I see it, there are 2 common use cases::  
- manual and optional  
- manual and required

The second state is the one where problems occur, because for a manual job, "required" can mean two different things:  
1. **The job** is required to run and succeed to proceed

(<https://www.facebook.com/gitlab>)

(<https://twitter.com/gitlab>)

(<https://www.linkedin.com/company/gitlab>)

(<https://www.youtube.com/channel/UCnMGQ8QHMAN>)

Sign up for GitLab's twice-monthly newsletter

abc@xyz.com

Subscribe

☐ I would like to receive the bi-weekly newsletter via email

Having trouble viewing or submitting this form?

Platform

Why GitLab (/why-gitlab/)

Pricing (/pricing/)

The DevOps Lifecycle (/stages-devops-lifecycle/)

Features (/features/)

Solutions

Enterprise (/enterprise/)

Small Business (/small-business/)

Continuous Integration (/solutions/delivery-automation/)

Resources

Blog (/blog/)

Install (/install/)

Docs (<https://docs.gitlab.com/>)

Developer Portal (<https://developer.gitlab.com/>)

Support

Get help (/get-help/)

Contact Sales (/sales/)

Support (/support/)

Status (<https://status.gitlab.com/>)

Company

About (/company/)

Jobs (/jobs/)

Leadership (/company/team/)

Hi there! What brings you to the site today?

Board of Directors (/company/team/board-of-directors/)

1

<https://about.gitlab.com/blog/2021/05/20/dag-manual-fix/>

7/8

Releases (/releases/)	Public Sector (/solutions/public-sector/)	Newsletter (/company/contact/)	Customers Portal (https://customers.gitlab.com/)	Team (/company/team/)
	Education (/education/)		Community Forum (https://forum.gitlab.com/)	Press (/press/)
				Investor Relations (https://ir.gitlab.com)
				Handbook (/handbook/)
				Terms of Use (/terms/)
				Privacy Policy (updated 6.10.22) (/privacy/)
				Préférences de cookies
				Shop (https://shop.gitlab.com/)

Git is a trademark of Software Freedom Conservancy and our use of 'GitLab' is under license

View [page source](#) — Edit in [Web IDE](#) — please [contribute \(https://gitlab.com/gitlab-com/www-gitlab-com/blob/master/CONTRIBUTING.md\)](https://gitlab.com/gitlab-com/www-gitlab-com/blob/master/CONTRIBUTING.md).

© 2022 GitLab B.V.

  
(https://twitter.com/gitlab)

  
(https://www.facebook.com/gitlab)

  
(https://www.youtube.com/channel/UCnMGQ8QHMANVIsI3xJrihhg)

  
(https://www.linkedin.com)

Hi there! What brings you to the site today?

1