needs

Version history

Use needs to execute jobs out-of-order. Relationships between jobs that use needs can be visualized as a directed acyclic graph.

You can ignore stage ordering and run some jobs without waiting for others to complete. Jobs in multiple stages can run concurrently.

Keyword type: Job keyword. You can use it only as part of a job.

Possible inputs:

- An array of jobs.
- An empty array ([]), to set the job to start as soon as the pipeline is created.

Example of needs:

```
linux:build:
  stage: build
  script: echo "Building linux..."

mac:build:
  stage: build
  script: echo "Building mac..."

lint:
  stage: test
  needs: []
  script: echo "Linting..."

linux:rspec:
  stage: test
  needs: ["linux:build"]
  script: echo "Running rspec on linux..."

mac:rspec:
  stage: test
  needs: ["mac:build"]
  script: echo "Running rspec on mac..."
```

```
production:
  stage: deploy
  script: echo "Running production..."
```

This example creates four paths of execution:

- Linter: The lint job runs immediately without waiting for the build stage to complete because it has no needs (needs: []).
- Linux path: The linux:rspec job runs as soon as the linux:build job finishes, without waiting for mac:build to finish.
- macOS path: The mac:rspec jobs runs as soon as the mac:build job finishes, without waiting for linux:build to finish.
- The production job runs as soon as all previous jobs finish: linux:build, linux:rspec, mac:build, mac:rspec.

Additional details:

- The maximum number of jobs that a single job can have in the needs array is limited:
  - For GitLab.com, the limit is 50. For more information, see our infrastructure issue.
  - For self-managed instances, the default limit is 50. This limit can be changed.
- If needs refers to a job that uses the parallel keyword, it depends on all jobs created in parallel, not just one job. It also downloads artifacts from all the parallel jobs by default. If the artifacts have the same name, they overwrite each other and only the last one downloaded is saved.
- In GitLab 14.1 and later you can refer to jobs in the same stage as the job you are configuring. This feature is enabled on GitLab.com and ready for production use. On self-managed GitLab 14.2 and later this feature is available by default.
- In GitLab 14.0 and older, you can only refer to jobs in earlier stages. Stages must be explicitly defined for all jobs that use the needs keyword, or are referenced in a job's needs section.
- In GitLab 13.9 and older, if needs refers to a job that might not be added to a pipeline because of only, except, or rules, the pipeline might fail to create.

needs:artifacts

Introduced in GitLab 12.6.

When a job uses needs, it no longer downloads all artifacts from previous stages by default, because jobs with needs can start before earlier stages complete. With needs you can only download artifacts from the jobs listed in the needs configuration.

Use artifacts: true (default) or artifacts: false to control when artifacts are downloaded in jobs that use needs.

Keyword type: Job keyword. You can use it only as part of a job. Must be used with needs:job.

Possible inputs:

- true (default) or false.

Example of needs:artifacts:

```
test-job1:
  stage: test
  needs:
    - job: build_job1
      artifacts: true

test-job2:
  stage: test
  needs:
    - job: build_job2
      artifacts: false

test-job3:
  needs:
    - job: build_job1
      artifacts: true
    - job: build_job2
    - build_job3
```

In this example:

- The test-job1 job downloads the build_job1 artifacts
- The test-job2 job does not download the build_job2 artifacts.

- The test-job3 job downloads the artifacts from all three build_jobs, because artifacts is true, or defaults to true, for all three needed jobs.

Additional details:

- In GitLab 12.6 and later, you can't combine the dependencies keyword with needs.

needs:project

premium

Introduced in GitLab 12.7.

Use needs:project to download artifacts from up to five jobs in other pipelines. The artifacts are downloaded from the latest successful pipeline for the specified ref.

If there is a pipeline running for the specified ref, a job with needs:project does not wait for the pipeline to complete. Instead, the job downloads the artifact from the latest pipeline that completed successfully.

needs:project must be used with job, ref, and artifacts.

Keyword type: Job keyword. You can use it only as part of a job.

Possible inputs:

- needs:project: A full project path, including namespace and group.
- job: The job to download artifacts from.
- ref: The ref to download artifacts from.
- artifacts: Must be true to download artifacts.

Examples of needs:project:

```
build_job:
  stage: build
  script:
    - ls -lhR
  needs:
    - project: namespace/group/project-name
```

```
    job: build-1
    ref: main
    artifacts: true
```

In this example, build_job downloads the artifacts from the latest successful build-1 job on the main branch in the group/project-name project.

In GitLab 13.3 and later, you can use CI/CD variables in needs:project, for example:

```
build_job:
  stage: build
  script:
    - ls -lhR
  needs:
    - project: $CI_PROJECT_PATH
      job: $DEPENDENCY_JOB_NAME
      ref: $ARTIFACTS_DOWNLOAD_REF
      artifacts: true
```

Additional details:

- To download artifacts from a different pipeline in the current project, set project to be the same as the current project, but use a different ref than the current pipeline. Concurrent pipelines running on the same ref could override the artifacts.
- The user running the pipeline must have at least the Reporter role for the group or project, or the group/project must have public visibility.
- You can't use needs:project in the same job as trigger.
- When using needs:project to download artifacts from another pipeline, the job does not wait for the needed job to complete. Directed acyclic graph behavior is limited to jobs in the same pipeline. Make sure that the needed job in the other pipeline completes before the job that needs it tries to download the artifacts.
- You can't download artifacts from jobs that run in parallel.
- Support for CI/CD variables in project, job, and ref was introduced in GitLab 13.3. Feature flag removed in GitLab 13.4.

Related topics:

- To download artifacts between parent-child pipelines, use needs:pipeline:job.

needs:pipeline:job

Introduced in GitLab 13.7.

A child pipeline can download artifacts from a job in its parent pipeline or another child pipeline in the same parent-child pipeline hierarchy.

Keyword type: Job keyword. You can use it only as part of a job.

Possible inputs:

- needs:pipeline: A pipeline ID. Must be a pipeline present in the same parent-child pipeline hierarchy.
- job: The job to download artifacts from.

Example of needs:pipeline:job:

- Parent pipeline (.gitlab-ci.yml):
- create-artifact:
- stage: build
- script: echo "sample artifact" > artifact.txt
- artifacts:
- paths: [artifact.txt]
-
- child-pipeline:
- stage: test
- trigger:
- include: child.yml
- strategy: depend
- variables:
- PARENT_PIPELINE_ID: $CI_PIPELINE_ID

Child pipeline (child.yml):

```
use-artifact:
  script: cat artifact.txt
  needs:
    - pipeline: $PARENT_PIPELINE_ID
```

```
    job: create-artifact
```

In this example, the create-artifact job in the parent pipeline creates some artifacts. The child-pipeline job triggers a child pipeline, and passes the CI_PIPELINE_ID variable to the child pipeline as a new PARENT_PIPELINE_ID variable. The child pipeline can use that variable in needs:pipeline to download artifacts from the parent pipeline.

Additional details:

- The pipeline attribute does not accept the current pipeline ID ($CI_PIPELINE_ID). To download artifacts from a job in the current pipeline, use needs.

needs:optional

Version history

To need a job that sometimes does not exist in the pipeline, add optional: true to the needs configuration. If not defined, optional: false is the default.

Jobs that use rules, only, or except might not always be added to a pipeline. GitLab checks the needs relationships before starting a pipeline:

- If the needs entry has optional: true and the needed job is present in the pipeline, the job waits for it to complete before starting.
- If the needed job is not present, the job can start when all other needs requirements are met.
- If the needs section contains only optional jobs, and none are added to the pipeline, the job starts immediately (the same as an empty needs entry: needs: []).
- If a needed job has optional: false, but it was not added to the pipeline, the pipeline fails to start with an error similar to: 'job1' job needs 'job2' job, but it was not added to the pipeline.

Keyword type: Job keyword. You can use it only as part of a job.

Example of needs:optional:

```
build-job:
  stage: build
```

```
test-job1:
  stage: test

test-job2:
  stage: test
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH

deploy-job:
  stage: deploy
  needs:
    - job: test-job2
      optional: true
    - job: test-job1

review-job:
  stage: deploy
  needs:
    - job: test-job2
      optional: true
```

In this example:

- build-job, test-job1, and test-job2 start in stage order.
- When the branch is the default branch, test-job2 is added to the pipeline, so:
  - deploy-job waits for both test-job1 and test-job2 to complete.
  - review-job waits for test-job2 to complete.
- When the branch is not the default branch, test-job2 is not added to the pipeline, so:
  - deploy-job waits for only test-job1 to complete, and does not wait for the missing test-job2.
  - review-job has no other needed jobs and starts immediately (at the same time as build-job), like needs: [].

needs:pipeline

You can mirror the pipeline status from an upstream pipeline to a bridge job by using the needs:pipeline keyword. The latest pipeline status from the default branch is replicated to the bridge job.

Keyword type: Job keyword. You can use it only as part of a job.

Possible inputs:

- A full project path, including namespace and group. If the project is in the same group or namespace, you can omit them from the project keyword. For example: project: group/project-name or project: project-name.

Example of needs:pipeline:

```
upstream_bridge:
  stage: test
  needs:
    pipeline: other/project
```

Additional details:

- If you add the job keyword to needs:pipeline, the job no longer mirrors the pipeline status. The behavior changes to needs:pipeline:job.