How to create Parent-Child Gitlab Pipeline

Table of contents

Reading Time: 4 minutes

GitLab parent-child pipeline run under the same get lab project. It can be useful when you want to run your pipeline under multiple conditions for example you may want to run your pipeline on merge requests, issue events, push events, on file changes, etc.

Problem with single global pipeline

Configuration for the single Global pipeline becomes very long and may become complicated and may be hard to manage in a single Global pipeline where all the steps in a stage must be completed before the next job begins. So this makes slowing things down in a pipeline.

And also it increases complexity, Reduces readability & not very flexible. You then need to write a lot of pipeline rules for different conditions for your pipeline.

Solution – Parent Child Pipeline

And to reduce complexity and increase the velocity of all steps in the stages then you should use the parent-child pipeline in GitLab that way you can reduce the complexity and increase the readability. Let's look at how to create a pipeline in GitLab.

Prerequisites

- Gitlab account & Repo
- Basic Understanding of Gitlab

Parent-Child Pipeline

Directory Structure

The directory structure for your application or pipeline repo should look like this. Command to check directory structure is tree -I .git -a

```
tree -I .git -a
```

The output will look like this:

```
.
├── app
│   └── index.html
├── .gitlab
│   ├── parent-pipe.yaml
│   └── workflows
│       ├── artifact-to-child.yaml
│       ├── build_app.yaml
│       └── merge-request-pipeline.yaml
├── .gitlab-ci.yml
└── README.md
```

.gitlab-ci.yml

The main Gitlab pipeline is defined in .gitlab-ci.yml file. It should be present at application's directory as shown in the directory structure.

Create a file named .gitlab-ci.yml & add the below content in this file.

```
## Details

# Author:           Rahul Soni ( rahul.soni@knoldus.com )
# Gitlab ID:        @Rahul-Soni28
# Github ID:        @Rahul-Soni28
# LinkedIn Profile: https://www.linkedin.com/in/rahul-soni-6592811b2/


## This will include content of local file into this file.
include:
  - local: .gitlab/parent-pipe.yaml
```

The **`include`** keyword is a list of objects or dictionaries. It includes the content of the file in the main .gitlab-ci.yml file.

It is including the parent pipeline code which we will be creating in the next step.

parent-pipe.yaml

Create a file called **parent-pipe.yaml** which will act as a parent pipeline & it runs in the first place.

Copy this content in the parent pipeline file.

```
stages:
  - artifacts
  - parent

## This job will generate artifacts and will send to artifact-to-child.yaml
child pipeline
artifacts:
  stage: artifacts
  variables:
    key1: value1
    key2: value2
  artifacts:
    paths:
      - ./artifacts.txt
  script:
    - echo "Generated Artifacts" > artifacts.txt

## This pipeline will only trigger when a MR
## is created.
merge_request_pipeline:
  stage: parent
  variables:
    pipeline: parent
    event: $CI_PIPELINE_SOURCE
  rules:
    - if: $CI_PIPELINE_SOURCE == "merge_request_event"
    - if: "$CI_MERGE_REQUEST_IID"
    - if: "$CI_COMMIT_TAG"
  trigger:
    include: .gitlab/workflows/merge-request-pipeline.yaml

    ## trigger:strategy to force the trigger job to wait for the downstream
pipeline to complete before it is marked as success.
    strategy: depend

## This job will run triggers a child which downloads artifacts generate in
previous jobs.
artifact-to-child:
  stage: parent
  variables:
    ## Needs: to make child download parent artifacts. ID of parent pipeline
needs to pass.
    PARENT_PIPELINE_ID: $CI_PIPELINE_ID
  trigger:
    include: .gitlab/workflows/artifact-to-child.yaml
    strategy: depend

## This job will trigger a child only when changes are made in particular
directory
build_app:
  stage: parent
  rules:
    - changes:
        - app/**/*
```

```
  trigger:
    include: .gitlab/workflows/build_app.yaml
    strategy: depend
```

Explaination: This pipeline contains two stages & 4 jobs. These Jobs can be controlled by `rules:` keyword to define when to run the job. These jobs act as parent jobs that trigger their child pipeline.

Child Pipelines

artifact-to-child.yaml

Create a file called **artifact-to-child.yaml** & copy this content in this file shown below.

```
stages:
  - build

artifact_job:
  stage: build
  needs:
    - pipeline: $PARENT_PIPELINE_ID
      job: artifacts
  script:
    - echo $CI_PIPELINE_SOURCE
    - echo $CI_COMMIT_TAG
    - ls
```

Explanation: This Job will receive the artifacts generated & received from the parent pipeline & parent job. With this artifact, further processes & jobs can be run in the child pipeline.

build_app.yaml

Create a file called **build_app.yaml** & copy the content in this file. This is simple child pipeline can be used to build your application. You can apply rules to change behavior for when to trigger this pipeline code.

```
stages:
  - build

app_build:
  stage: build
  script:
    - echo $CI_PIPELINE_SOURCE
    - echo $CI_COMMIT_TAG
    - ls
```

merge-request-pipeline.yaml

Create a file called **merge-request-pipeline.yaml**. This is merge request child pipeline which will run every time someone creates or updates a merge request.

```
workflow:
  rules:
    - if: "$CI_MERGE_REQUEST_IID"
    - if: "$CI_COMMIT_TAG"
stages:
  - merge

merge_request:
  stage: merge
  variables:
    name: rahul

  script:
    - echo $name
    - echo "$pipeline triggred this child when $event event is created"
    - date
    - echo $CI_PIPELINE_SOURCE
    - echo $CI_COMMIT_TAG
    - echo $CI_COMMIT_BRANCH
    - echo "$CI_MERGE_REQUEST_IID Merge Request created from
\"$CI_MERGE_REQUEST_SOURCE_BRANCH_NAME\" branch --->
\"$CI_MERGE_REQUEST_TARGET_BRANCH_NAME\" branch"
```