* A build stage, with a job called compile.
* A test stage, with two jobs called test1 and test2.
* A staging stage, with a job called deploy-to-stage.
* A production stage, with a job called deploy-to-prod.

**Types of pipelines**

* [Basic pipelines](https://docs.gitlab.com/ee/ci/pipelines/pipeline_architectures.html#basic-pipelines) run everything in each stage concurrently, followed by the next stage.
* [Directed Acyclic Graph Pipeline (DAG) pipelines](https://docs.gitlab.com/ee/ci/directed_acyclic_graph/index.html) are based on relationships between jobs and can run more quickly than basic pipelines.
* [Multi-project pipelines](https://docs.gitlab.com/ee/ci/multi_project_pipelines.html) combine pipelines for different projects together.
* [Parent-Child pipelines](https://docs.gitlab.com/ee/ci/parent_child_pipelines.html) break down complex pipelines into one parent pipeline that can trigger multiple child sub-pipelines, which all run in the same project and with the same SHA.
* [Pipelines for Merge Requests](https://docs.gitlab.com/ee/ci/merge_request_pipelines/index.html) run for merge requests only (rather than for every commit).
* [Pipelines for Merged Results](https://docs.gitlab.com/ee/ci/merge_request_pipelines/pipelines_for_merged_results/index.html) are merge request pipelines that act as though the changes from the source branch have already been merged into the target branch.
* [Merge Trains](https://docs.gitlab.com/ee/ci/merge_request_pipelines/pipelines_for_merged_results/merge_trains/index.html) use pipelines for merged results to queue merges one after the other.

1. Jobs are the basic component of the pipeline
2. Stages are defined by the **stage** keyword.

We can view pipeline CI/CD> Pipeline

From the same page we can retry filed jobs.

To execute a pipeline manually:

1. Navigate to your project’s **CI/CD > Pipelines**.
2. Click on the **Run Pipeline** button.
3. On the **Run Pipeline** page:
   1. Select the branch to run the pipeline for in the **Create for** field.
   2. Enter any [environment variables](https://docs.gitlab.com/ee/ci/variables/README.html) required for the pipeline run.
   3. Click the **Create pipeline** button.

We can run a pipeline with the URL string

For example, the query string .../pipelines/new?ref=my\_branch&var[foo]=bar&file\_var[file\_foo]=file\_bar will pre-populate the **Run Pipeline** page with:

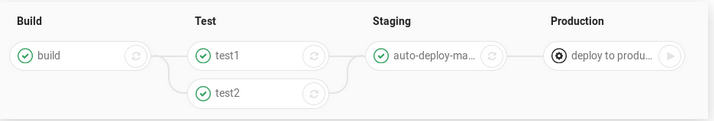
* **Run for** field: my\_branch.
* **Variables** section:
  + Variable:
    - Key: foo
    - Value: bar
  + File:
    - Key: file\_foo
    - Value: file\_bar

The format of the pipelines/new URL is:

.../pipelines/new?ref=<branch>&var[<variable\_key>]=<value>&file\_var[<file\_key>]=<value>

**Add Manual interaction to your pipeline**

Manual action configured using the **when:manual** parameter allow you to require manual interaction before moving forward in the pipeline.



Here in this case manual interaction required.

We can start multiple manual action in stages by clicking the play all manual.

But it only available to the 1. Who have developer access 2. If the stage contains manual actions.

**Delete Pipeline**

Users with owner permission can delete the pipeline.

Deleting a pipeline action cannot be undone

**Pipeline security on protected branches:**

The following actions are allowed on protected branches only if the user is [allowed to merge or push](https://docs.gitlab.com/ee/user/project/protected_branches.html#using-the-allowed-to-merge-and-allowed-to-push-settings) on that specific branch:

* Run manual pipelines (using the [Web UI](https://docs.gitlab.com/ee/ci/pipelines/#run-a-pipeline-manually) or [pipelines API](https://docs.gitlab.com/ee/ci/pipelines/#pipelines-api)).
* Run scheduled pipelines.
* Run pipelines using triggers.
* Trigger manual actions on existing pipelines.
* Retry or cancel existing jobs (using the Web UI or pipelines API).

Variables marked as protected are accessible only to the protected branches.

Runners marked as protected can run jobs only protected branches.

The detailed job status can be found on view job status.

**Specifiying variables when running munal jobs:**

When running manual jobs, you can supply additional job specific variables.

You can do this from the job page of the manual job you want to run with additional variables.

to access this page click on the name of the job.

This is useful when you want to alter the execution of job that uses custom variables, adding a new variable here override the value defined in the UI or .gitlab-ci.yml for a single run.

**Delay a Job**

When you do not want to run a job immediately, you can use when:delayed parameter to delay a job’s execution for a certain period.