

GitLab Pipelines for Every Need

Debsankha Manik

MPI for Dynamics and Self-Organization, Göttingen

July 3, 2019

CI/CD Pipelines: Core Idea

Apply well defined operations on the codebase automatically

CI/CD Pipelines: Core Idea

Apply well defined operations on the codebase automatically

On each commit (or according to some fine-tuned criteria)

CI/CD Pipelines: Core Idea

Apply well defined operations on the codebase automatically

On each commit (or according to some fine-tuned criteria)

- 1 A pre-specified environment is created (typically using `docker`).

CI/CD Pipelines: Core Idea

Apply well defined operations on the codebase automatically

On each commit (or according to some fine-tuned criteria)

- 1 A pre-specified environment is created (typically using `docker`).
- 2 A pre-defined set of operations are run on the codebase.

CI/CD Pipelines: Core Idea

Apply well defined operations on the codebase automatically

On each commit (or according to some fine-tuned criteria)

- 1 A pre-specified environment is created (typically using `docker`).
- 2 A pre-defined set of operations are run on the codebase.
- 3 Operations may depend on each other, be chronologically ordered, asked to run on parallel ...

Use Case 1: Automated Testing

Let's say a codebase has a comprehensive test suite.

Use Case 1: Automated Testing

Let's say a codebase has a comprehensive test suite.

But new team member cannot run it.

Use Case 1: Automated Testing

Let's say a codebase has a comprehensive test suite.

But new team member cannot run it.

Because some tests depend on library `foo` being compiled with `bar=True` option.

Use Case 1: Automated Testing

Let's say a codebase has a comprehensive test suite.

But new team member cannot run it.

Because some tests depend on library `foo` being compiled with `bar=True` option.

Nobody knows whether the codebase is deployable or not.

Use Case 1: Automated Testing

Let's say a codebase has a comprehensive test suite.

But new team member cannot run it.

Because some tests depend on library `foo` being compiled with `bar=True` option.

Nobody knows whether the codebase is deployable or not.

What if we:

Use Case 1: Automated Testing

Let's say a codebase has a comprehensive test suite.

But new team member cannot run it.

Because some tests depend on library `foo` being compiled with `bar=True` option.

Nobody knows whether the codebase is deployable or not.

What if we:

1. Ran our test suite on exactly defined environment on each commit.

Use Case 1: Automated Testing

Let's say a codebase has a comprehensive test suite.

But new team member cannot run it.

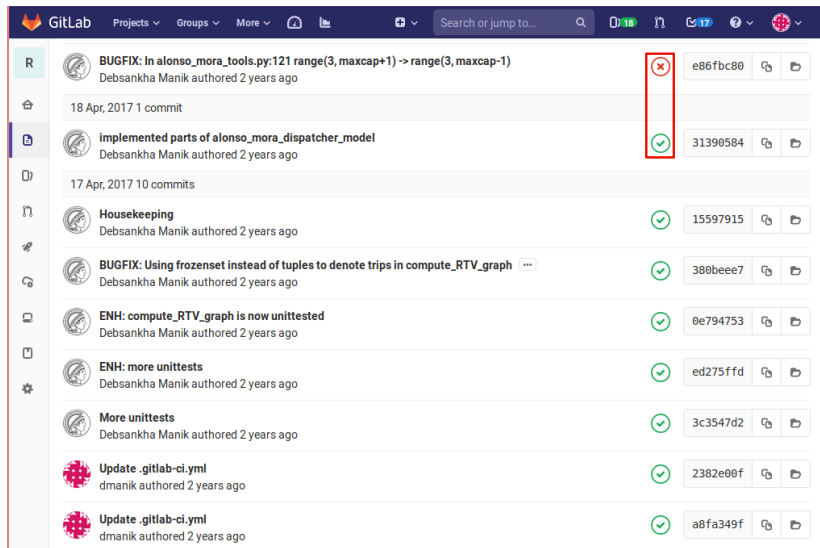
Because some tests depend on library `foo` being compiled with `bar=True` option.

Nobody knows whether the codebase is deployable or not.

What if we:

1. Ran our test suite on exactly defined environment on each commit.
2. Could see which commit caused the test suite to fail.

We Can Have Exactly That



The screenshot displays the GitLab interface with a list of commits. The top navigation bar includes the GitLab logo, 'Projects', 'Groups', 'More', a search bar, and notification icons. The commit list is as follows:

Commit Hash	Author	Date	Commits	Status
e86fbc80	Debsankha Manik	18 Apr, 2017	1 commit	❌
31390584	Debsankha Manik	17 Apr, 2017	10 commits	✅
15597915	Debsankha Manik	2 years ago		✅
380bee7	Debsankha Manik	2 years ago		✅
0e794753	Debsankha Manik	2 years ago		✅
ed275ffd	Debsankha Manik	2 years ago		✅
3c3547d2	Debsankha Manik	2 years ago		✅
2382e00f	dmanik	2 years ago		✅
a8fa349f	dmanik	2 years ago		✅

Badge screaming loudly if `master` is broken

The screenshot shows the GitLab interface for the repository 'Ecoptimizers' on the 'Ecoptimizer' branch. The commit hash is 'cfe84fa4', authored 5 months ago. The commit message is 'trying custom docker image to eliminate slow apt-get install'. Below the commit information is a table of files in the repository. At the bottom of the page, a badge for the pipeline is shown with the text 'pipeline failed', which is highlighted with a red box.

Ecoptimizers > Ecoptimizer > Repository

gitlab-ci Ecoptimizer / +

History Find file Web IDE

trying custom docker image to eliminate slow apt-get install cfe84fa4

dmanik authored 5 months ago

Name	Last commit	Last update
.git-crypt	Add 1 git-crypt collaborator	10 months ago
ecoptimizer	Merge branch 'master' of gitlab.gwdg.de:Ecoptimizers/Ecopti...	5 months ago
.gitignore	sqlite support for wheelchair, bike, stroller type loads and giti...	10 months ago
.gitlab-ci.yml	trying custom docker image to eliminate slow apt-get install	5 months ago
README.md	Continuous integration successfully implemented.	5 months ago
setup.py	fix typo setup.py	5 months ago

README.md

pipeline failed

How: Use a CI/CD Pipeline to Run the Tests

- 1 Choose docker image with base dependencies of your software.

How: Use a CI/CD Pipeline to Run the Tests

- 1 Choose docker image with base dependencies of your software.
- 2 Install additional dependencies using package manager e.g. `apt` , if needed.

How: Use a CI/CD Pipeline to Run the Tests

- 1 Choose docker image with base dependencies of your software.
- 2 Install additional dependencies using package manager e.g. `apt`, if needed.
- 3 Run the test suite using whatever test runner you want.

How: Use a CI/CD Pipeline to Run the Tests

- 1 Choose docker image with base dependencies of your software.
- 2 Install additional dependencies using package manager e.g. `apt`, if needed.
- 3 Run the test suite using whatever test runner you want.
 - So long the shell gets a 0 return code on success and nonzero on failure.

How to Set This Up?



Use Case 2: Integration Tests

We had some tests that “simulates” our deployed system, i.e.

- ① Installs all the components of our software.
- ② Spins up the server.
- ③ Creates a few clients.
- ④ The clients fire a few hundred requests.
- ⑤ We check if the server was able to handle all these requests, satisfying some predefined criteria.

Use Case 2: Integration Tests

We had some tests that “simulates” our deployed system, i.e.

- ① Installs all the components of our software.
- ② Spins up the server.
- ③ Creates a few clients.
- ④ The clients fire a few hundred requests.
- ⑤ We check if the server was able to handle all these requests, satisfying some predefined criteria.

Running these tests didn't fit in the basic CI/CD paradigm:

- Takes too long to run.
- We didn't need this to run on every commit to every branch.

Use Case 2: Integration Tests

We had some tests that “simulates” our deployed system, i.e.

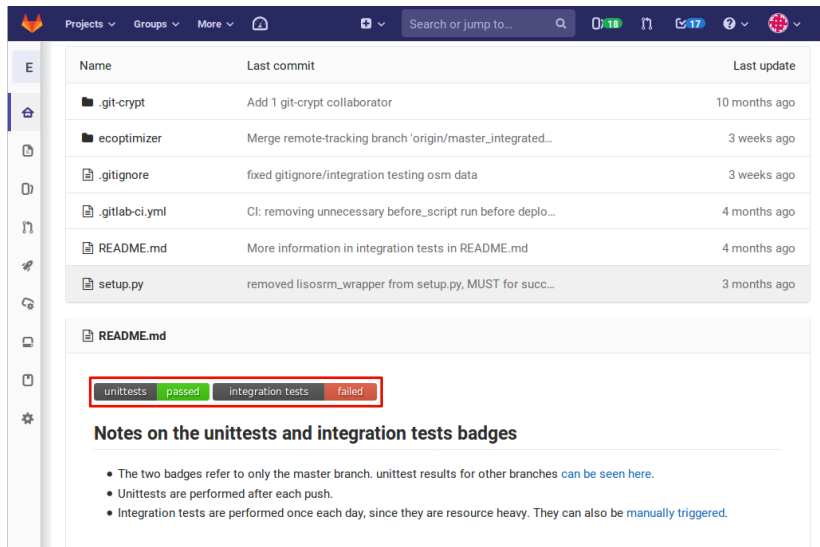
- ① Installs all the components of our software.
- ② Spins up the server.
- ③ Creates a few clients.
- ④ The clients fire a few hundred requests.
- ⑤ We check if the server was able to handle all these requests, satisfying some predefined criteria.

Running these tests didn't fit in the basic CI/CD paradigm:







- Takes too long to run.
- We didn't need this to run on every commit to every branch.

We wanted a nightly run of these “integration tests” on master.

We Ended Up With This



The screenshot shows the GitLab interface for a project. The top navigation bar includes 'Projects', 'Groups', 'More', a search bar, and notification icons. The left sidebar contains navigation icons for home, search, and settings. The main content area displays a table of files and folders with their last commit and update dates.

Name	Last commit	Last update
 .git-crypt	Add 1 git-crypt collaborator	10 months ago
 ecoptimizer	Merge remote-tracking branch 'origin/master_integrated...	3 weeks ago
 .gitignore	fixed gitignore/integration testing osm data	3 weeks ago
 .gitlab-ci.yml	CI: removing unnecessary before_script run before deplo...	4 months ago
 README.md	More information in integration tests in README.md	4 months ago
 setup.py	removed lisosrm_wrapper from setup.py, MUST for succ...	3 months ago

Below the file list, there is a section for the README.md file. It features two test badges: 'unittests passed' (green) and 'integration tests failed' (red). A red box highlights these two badges.

Notes on the unittests and integration tests badges

- The two badges refer to only the master branch. unittest results for other branches [can be seen here](#).
- Unittests are performed after each push.
- Integration tests are performed once each day, since they are resource heavy. They can also be [manually triggered](#).

How to Set This Up?

Specify an environment as before using `docker`.

How to Set This Up?

Specify an environment as before using `docker`.

Specify a new `stage` in the CI pipeline that is to be triggered only on schedule, not automatically on each commit.

How to Set This Up?

Specify an environment as before using `docker`.

Specify a new `stage` in the CI pipeline that is to be triggered only on schedule, not automatically on each commit.

How to Set This Up?

Specify an environment as before using `docker`.

Specify a new `stage` in the CI pipeline that is to be triggered only on schedule, not automatically on each commit.

Generate custom badges displaying this stage passes/fails using `artifact s` (more on this later).

How to Set This Up?

Specify an environment as before using `docker`.

Specify a new `stage` in the CI pipeline that is to be triggered only on schedule, not automatically on each commit.

Generate custom badges displaying this stage passes/fails using `artifact s` (more on this later).

Use Case 3: ~~Writing a Paper~~ Making This Presentation

<https://gitlab.gwdg.de/dmanik/gitlab-ci-talk>

The screenshot shows the GitLab interface for the repository 'dmanik/gitlab-ci-talk'. The top navigation bar includes 'Projects', 'Groups', and 'More' menus, a search bar, and notification icons. The left sidebar contains navigation icons for home, repository, and settings. The main content area displays a table of files and folders with columns for 'Name', 'Last commit', and 'Last update'. Below the table, the 'README.md' file is selected, showing a commit message and a status indicator 'pipeline passed' highlighted with a red box. Below the status indicator is a link to 'Grab the freshest PDF.'.

Name	Last commit	Last update
figures	talk almost complete, some fleshing out to do ...	14 hours ago
snippets	slides for scenario-1 done, added figures as well	1 day ago
.gitignore	Updates notes: maybe talk about merge reque...	5 days ago
.gitlab-ci.yml	Added figure of pipelines view on GitLab	5 days ago
NOTES.md	slides for scenario-1 done, added figures as well	1 day ago
README.md	fixed link to pdf in README	5 days ago
talk.tex	talk almost complete, some fleshing out to do ...	14 hours ago

README.md

A talk on GitLab CI/CD pipelines

pipeline passed

Grab the [freshest PDF.](#)

How to Set This Up?

Use a docker image with `texlive` (here `blang/latex`)

How to Set This Up?

Use a docker image with `texlive` (here `blang/latex`)

Define a job that invokes `latexmk` to compile the PDF.

How to Set This Up?

Use a docker image with `texlive` (here `blang/latex`)

Define a job that invokes `latexmk` to compile the PDF.

How to Set This Up?

Use a docker image with `texlive` (here `blang/latex`)

Define a job that invokes `latexmk` to compile the PDF.

Use `job artifacts` to store the compiled PDF.

How to Set This Up?

Use a docker image with `texlive` (here `blang/latex`)

Define a job that invokes `latexmk` to compile the PDF.

Use `job artifacts` to store the compiled PDF.

How to Set This Up?

Use a docker image with `texlive` (here `blang/latex`)

Define a job that invokes `latexmk` to compile the PDF.

Use `job artifacts` to store the compiled PDF.

We ask GitLab to store all files matching the wildcard `*.pdf`.

How to Set This Up?

Use a docker image with `texlive` (here `blang/latex`)

Define a job that invokes `latexmk` to compile the PDF.

Use `job artifacts` to store the compiled PDF.

We ask GitLab to store all files matching the wildcard `*.pdf`.

The PDF is then accessible under the URL

`<repo_root>/-/jobs/artifacts/master/raw/talk.pdf?job=makepdf`.

Outlook

Outlook

Multi project pipelines (build/test the full stack divided across repositories).

Outlook

Multi project pipelines (build/test the full stack divided across repositories).

Automatically run test suite on merge requests.

- No need to waste time reviewing a MR if the test suite do not pass.

Outlook

Multi project pipelines (build/test the full stack divided across repositories).

Automatically run test suite on merge requests.

- No need to waste time reviewing a MR if the test suite do not pass.

Need to download large amounts of data for tests to run: use `cache`, maybe docker `volumes`?

Outlook

Multi project pipelines (build/test the full stack divided across repositories).

Automatically run test suite on merge requests.

- No need to waste time reviewing a MR if the test suite do not pass.

Need to download large amounts of data for tests to run: use `cache`, maybe docker `volumes`?

Skipping the pipeline on certain commits (`except` keyword in CI config).

- e.g. changing the `README`.

Outlook

Multi project pipelines (build/test the full stack divided across repositories).

Automatically run test suite on merge requests.

- No need to waste time reviewing a MR if the test suite do not pass.

Need to download large amounts of data for tests to run: use `cache`, maybe docker `volumes`?

Skipping the pipeline on certain commits (`except` keyword in CI config).

- e.g. changing the `README`.

Publish your package to PyPi, deploy to staging/production.

Outlook

Multi project pipelines (build/test the full stack divided across repositories).

Automatically run test suite on merge requests.

- No need to waste time reviewing a MR if the test suite do not pass.

Need to download large amounts of data for tests to run: use `cache`, maybe docker `volumes`?

Skipping the pipeline on certain commits (`except` keyword in CI config).

- e.g. changing the `README`.

Publish your package to PyPi, deploy to staging/production.

Reproducible papers?