UNIT 7.
CONTINUOUS INTEGRATION

Web Applications Deployment
CFGS DAW

Author: Carlos Cacho López

Reviewed by: Lionel Tarazón Alcocer

lionel.tarazon@ceedcv.es

2019/2020

License

Attribution – NonCommercial – ShareAlike (by-nc-sa): You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may not use the material for commercial purposes. If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

Nomenclature

During this unit we are going to use special symbols to distinct some important elements.
This symbols are:

 Important

 Attention

 Interesting

INDEX

UD07. CONTINUOUS INTEGRATION

1. INTRODUCTION

Consider an application that has its code stored in a Git repository in GitLab. Developers push code changes every day, multiple times a day. For every push to the repository, you can create a set of scripts to build and test your application automatically, decreasing the chance of introducing errors to your app.

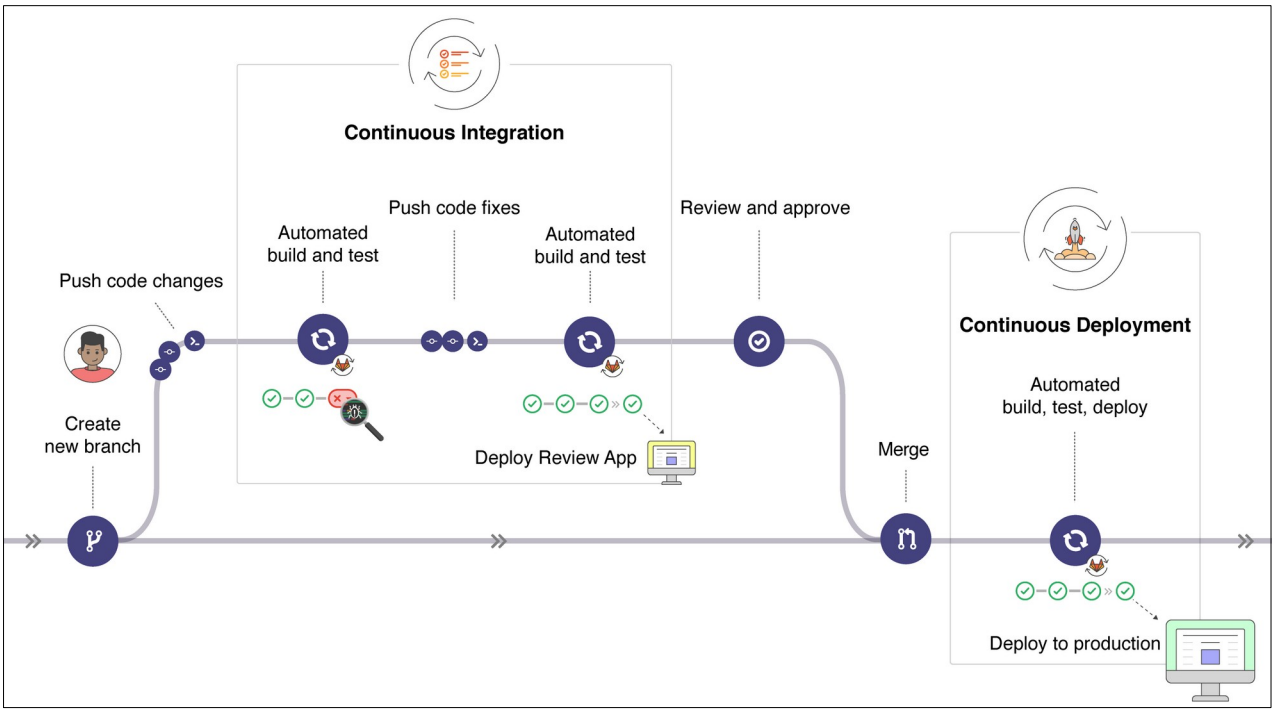
This practice is known as **Continuous Integration (CI)**; for every change submitted to an application it's built and tested automatically and continuously, ensuring the introduced changes pass all tests, guidelines, and code compliance standards you established for your app.

Others concept very close to CI are:

- **Continuous Delivery (CD)** is a step beyond Continuous Integration. Your application is not only built and tested at every code change pushed to the codebase, but, as an additional step, it's also deployed continuously, though the deployments are triggered manually. This method ensures the code is checked automatically but requires human intervention to manually and strategically trigger the deployment of the changes.

- **Continuous Deployment (CD)** is also a further step beyond Continuous Integration, similar to Continuous Delivery. The difference is that instead of deploying your application manually, you set it to be deployed automatically. It does not require human intervention at all to have your application deployed.

Here we can see a diagram with the basic CI/CD workflow:



From https://docs.gitlab.com/ee/ci licensed under CC BY-SA 4.0

2. ADVANTAGES

Some advantages of the CI are:

- Integration bugs are detected early and are easy to find them due to small change sets.
- Avoids last-minute chaos at release dates.
- Developers need to perform a small number of changes to revert the codebase to a bug-free state when unit tests fail.
- Constant availability of a "current" build for testing, demo, or release purposes.
- Frequent code check-in pushes developers to create modular, less complex code.
- Enforces discipline of frequent automated testing.
- Immediate feedback on system-wide impact of local changes.

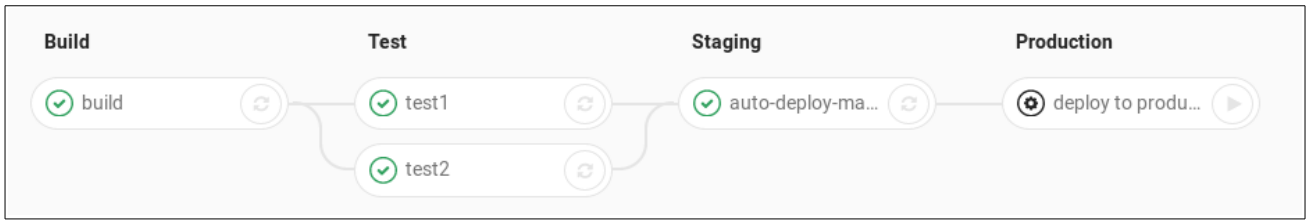
The main disadvantage is the mount of work involved in setting and configuring propper automated tests as well as setting up abuild system.

3. PIPELINES

Adapted from https://docs.gitlab.com/ee/ci/pipelines.html licensed under CC BY-SA 4.0

A pipeline is a group of jobs that get executed in stages. All of the jobs in a stage are run in parallel, and if they all succeed the pipeline moves on to the next stage. We can define all the stages we need.

Here can see four stages: Build, Test, Staging and Production.

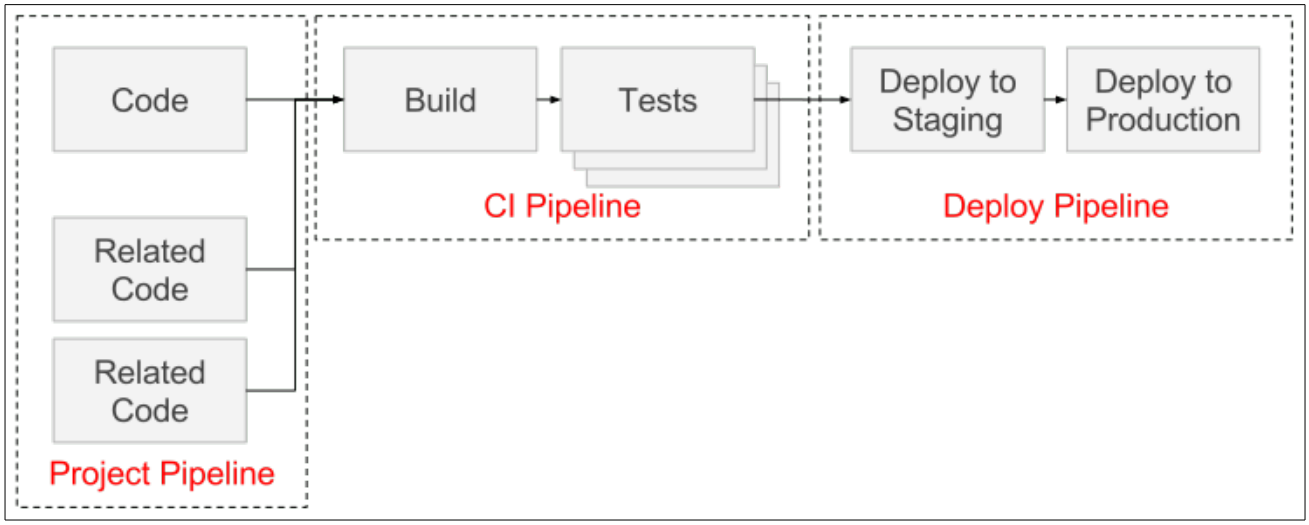


The process starts after a git push to the repository:

1. First, the *Build* job is executed.
2. If the *Build* job succeeds, the *Test* jobs (test1 and test2) are run in parallel.
3. If all *Test* jobs succeed, the *Staging* job is run.
4. If the *Staging* job succeeds, the *Production* job is run.

If the *Production* job succeeds, the commit is marked as *passed*.
If any of the jobs fail, the commit is marked as *failed* and no jobs of further stage are run.

There are three types of pipelines that often use the single shorthand of “pipeline”. People often talk about them as if each one is “the” pipeline, but really, they’re just pieces of a single, comprehensive pipeline:



4. BIBLIOGRAPHY

[1] *https://docs.gitlab.com/ee/ci/*

[2] https://www.atlassian.com/continuous-delivery