Continuous Integration

James Quilty

School of Engineering and Computer Science
Victoria University of Wellington

Introduction to Continuous Integration

“Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily — leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.”

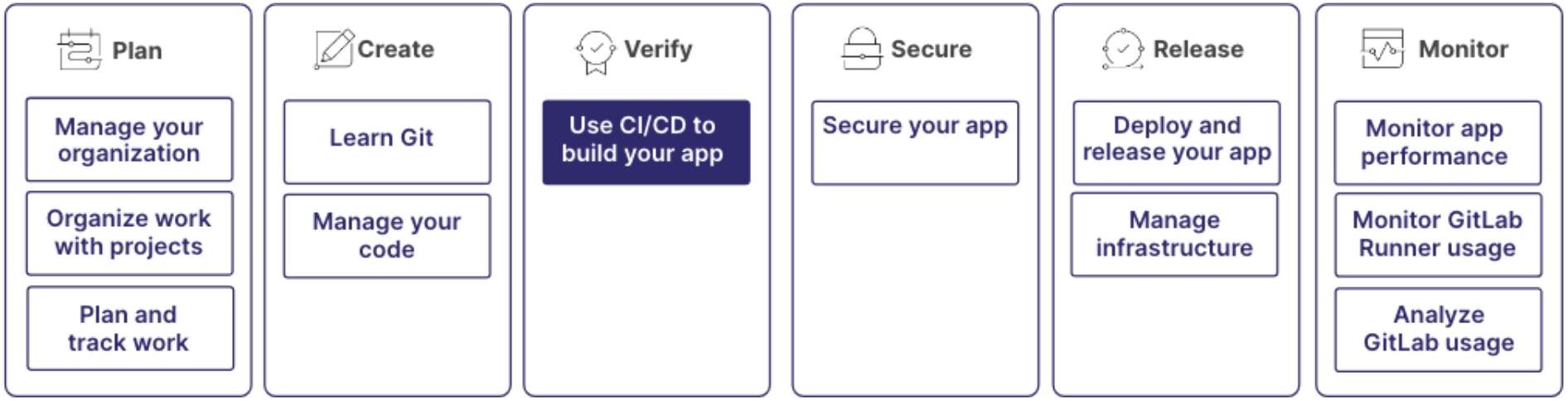
–Martin Fowler, 2006

`https://martinfowler.com/articles/continuousIntegration.html`

This lecture: theory & practice with examples in GitLab’s *CI pipeline*.

Introduction to Continuous Integration

“CI/CD is a continuous method of software development, where you continuously build, test, deploy, and monitor iterative code changes.



“This process is part of a larger workflow:”

`https://docs.gitlab.com/ee/ci/introduction/index.html`

Key Concepts

Continuous Integration is:

- a *practice*;
- based on *individual* contributions;
- in the context of a *group*.

Continuous Integration is not:

- a technology;
- automation;
- sub-atomic contributions.

Case Study

So, a student was working on a *really* valuable feature:

- they made the changes in their working copy;
- they built their not-quite-complete work;
- they uploaded it to their DR;
- the code appeared to be working;
- their lab session ended and they left. . .

Case Study

So, a student was working on a *really* valuable feature:

- they made the changes in their working copy;
- they built their not-quite-complete work;
- they uploaded it to their DR;
- the code appeared to be working;
- their lab session ended and they left. . .
- another team started development on the DR;
- they found a blocking bug in the code on the DR;
- they wanted the feature but couldn't find the code in GitLab;
- they weren't able to fix the bug and benefit from the feature. . .
- until the first student integrated their work.

Characteristics of Continuous Integration

Integration and Continuous Integration are characterised by:

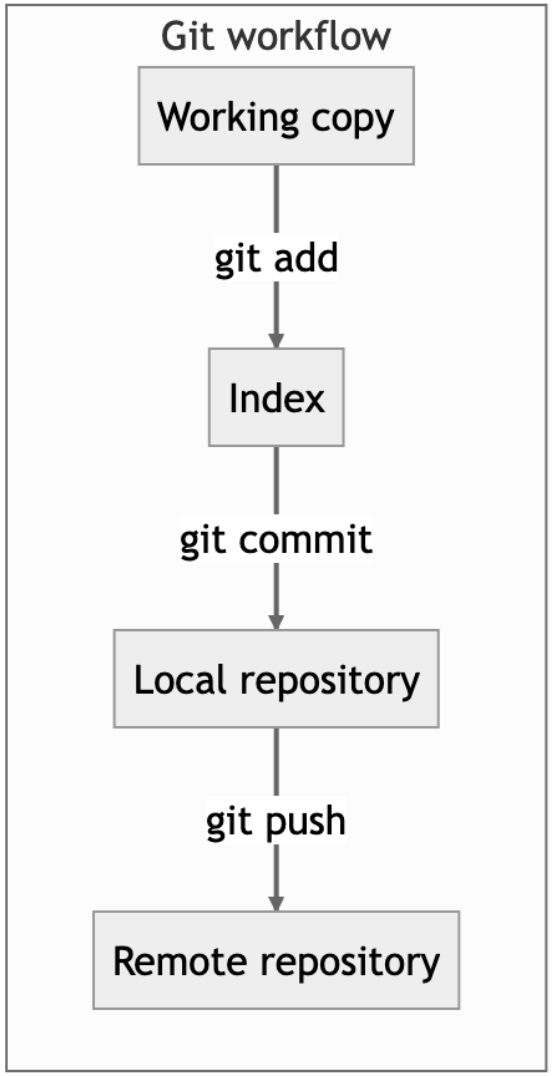
- sharing *work in progress*;
- contributing *partial* or *incomplete* work;
- sharing work *even when you're not completely happy with it*;

In this way, you make most effective use of the *source control management* and *project management* tools.

Recap: Using git's workflow Effectively

Guidelines (Lecture 8)

- Greatest freedom at the start, least freedom at the end
- Use staging to collect changes for a commit *while working*
- Local commits are easy to change, both content and order
- Remote is *shared* and commits should be changed *with caution*

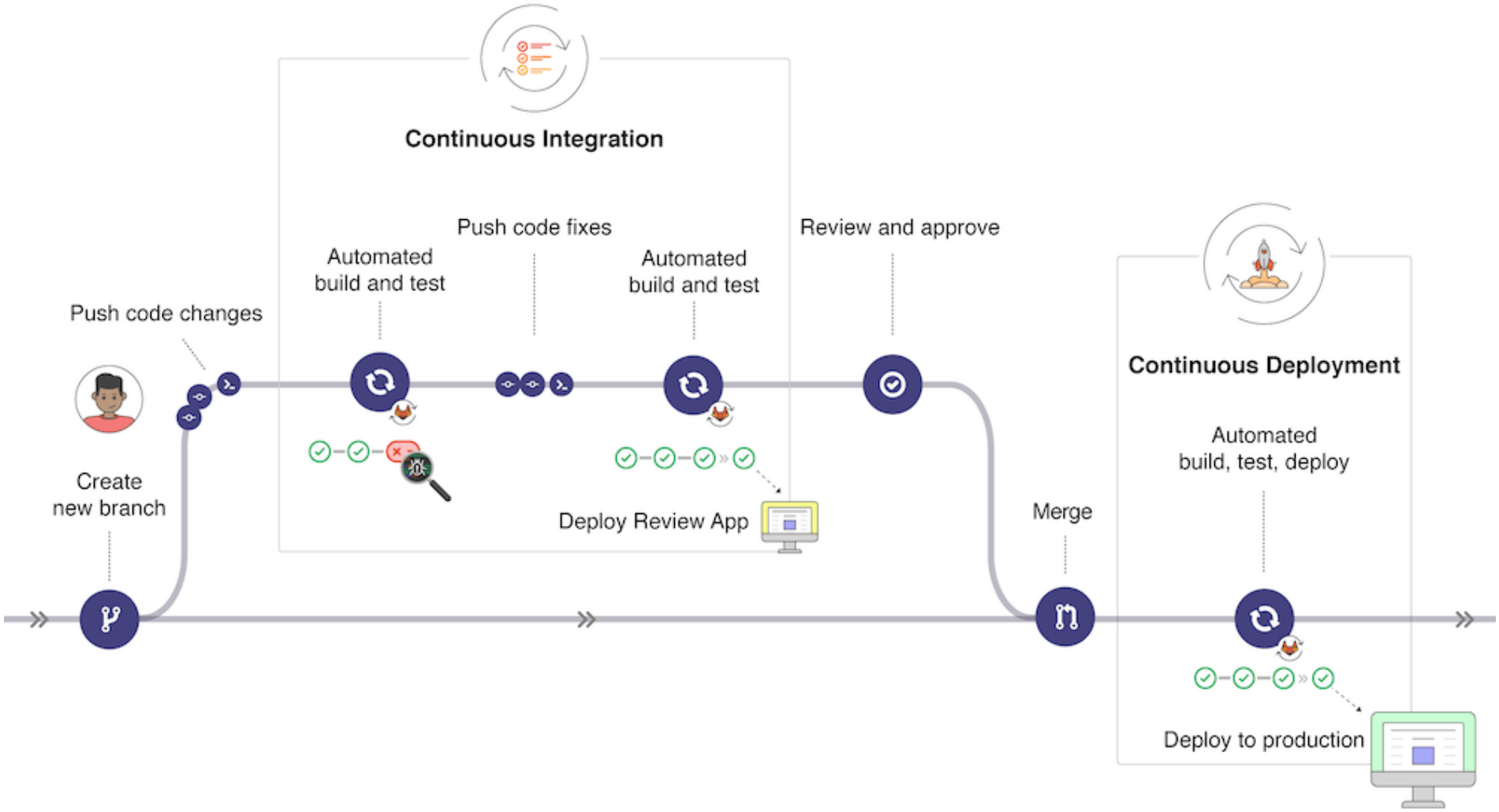


Essential Practices

From *Continuous Delivery* by Jez Humble and David Farley:

1. Don’t Push to a Broken Build;
2. Wait for Commit Tests to Pass before Moving On
3. Always Run All Commit Tests Locally before Committing
4. Never Go Home on a Broken Build
5. Always Be Prepared to Revert to the Previous Revision
6. Time-Box Fixing before Reverting
7. Don’t Comment-out Failing Tests
8. Take Responsibility for Your Breakages
9. Follow Test-Driven Development

GitLab Workflow Example



https://docs.gitlab.com/ee/ci/

Continuous Integration Automation

GitLab implementation:

- Pipelines `https://docs.gitlab.com/ee/ci/pipelines`
- Job Artefacts `https://bit.ly/4b163N5`
- Variables `https://docs.gitlab.com/ee/ci/variables/`
- Caching `https://docs.gitlab.com/ee/ci/caching/`
- GitLab Runner `https://docs.gitlab.com/runner/`
- Container registry `https://bit.ly/4b8GKIQ`

Let's take a look at how these are used in the Data Recorder's CI pipeline. . .

Example: Data recorder CI

How does the CI pipeline function in the Data Recorder project?

- `https://gitlab.ecs.vuw.ac.nz/course-work/engr301/2024/templates/data-recorder`
- `https://gitlab.ecs.vuw.ac.nz/course-work/engr301/2024/templates/data-recorder/-/blob/main/.gitlab-ci.yml`
- `https://gitlab.ecs.vuw.ac.nz/course-work/engr301/2024/templates/data-recorder/-/pipelines`

Electronics

Electronics needs continuous integration as much as any other area of the project!

- Branching strategy
- Bill of Materials (BOM) management
- Automatic generation of schematic and layout PDFs
- Automatic generation of files for manufacture

KiBot `https://github.com/INTI-CMNB/KiBot`