COURSEWARE

Continuous Integration

Contents

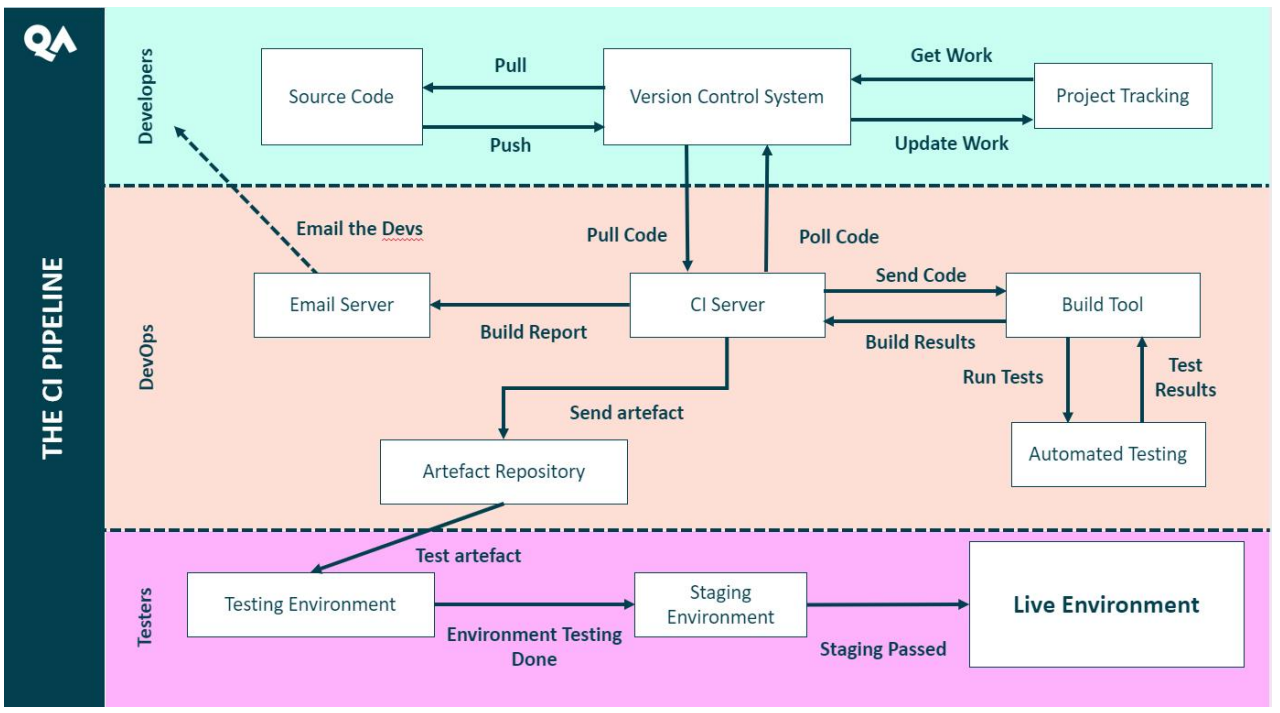
Overview

Continuous Integration (CI) refers to the automated integration of code from many contributors into a single software project.

The purpose of the CI pipeline approach is to allow developers to integrate newly-generated code *easily* and *frequently*, and is achieved through the use of automated testing tools to check the correctness of code before full integration. Additional checks that may also be performed include syntax style, code quality, etc.

At the heart of the process is the **version control system** (VCS) and the **CI server**.

- The VCS is designed to track changes to code over time as contributors add new features to the application. This system allows for cohesive collaboration and the ability to easily revert an application to a previous, stable state if new code breaks something.
- The CI server handles all the automated building, testing, and deployment of code as it is pushed to the VCS.



This pipeline diagram illustrates the many parts that make up the CI pipeline. It is complex and may be difficult to decipher at first, so this module breaks things down to help simplify each stage.

What CI does

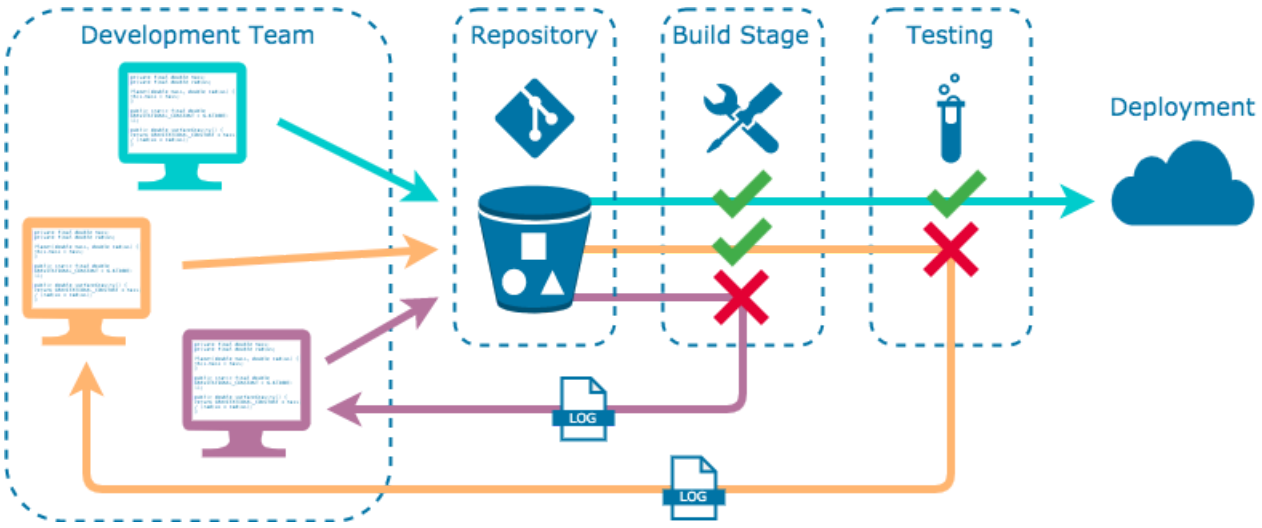
CI allows developers to generate and implement new functionality with ease and speed through automation of the integration process. Building and testing of new code is handled through automation, allowing developers to focus their efforts on the creation of new features.

Developers work on features independently and in parallel with each other, without having to worry about clashes within their teams. Tracking and merging code additions allows for smoother integration of new features without fear of irreparably damaging the source code.

A continuous integration pipeline should:

- Maintain a single source code repository for a project
- Have a "master" branch that should **always** be ready to deploy
- Keep all team members informed of every update to the source code
- Automate build processes
- Automate testing of new builds
- Inform developers of test failures with detailed logs
- Encourage smaller, frequent deployments of code

How CI works

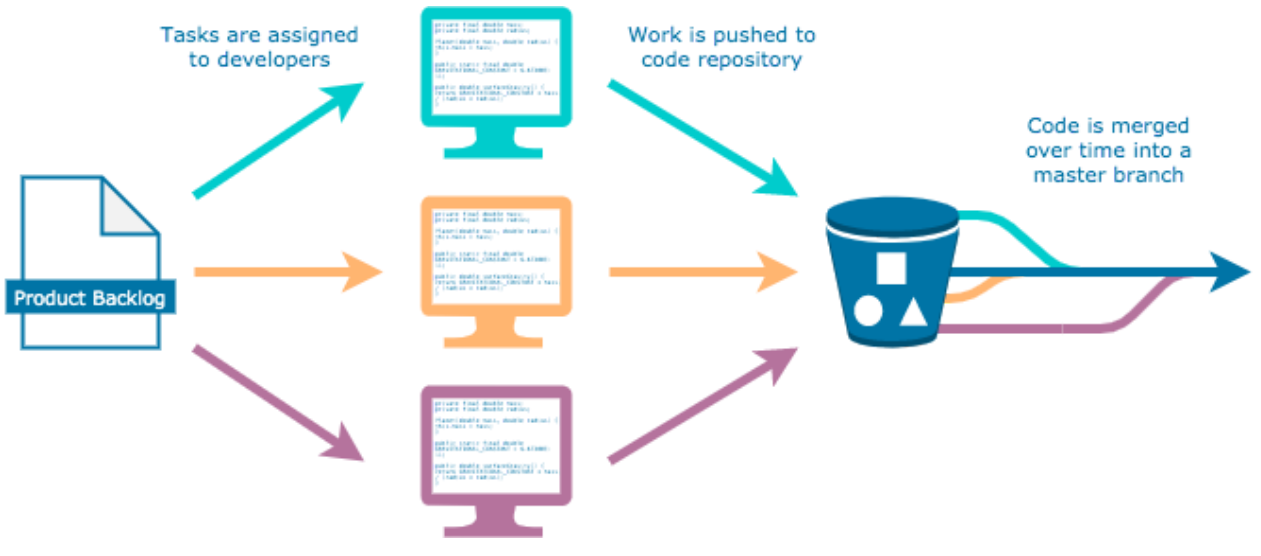


To achieve this pipeline, CI leverages many software tools to handle the automated building, testing, and deployment processes. The main steps in the CI pipeline include:

Code Generation

Typically a CI pipeline would be utilised within an agile workflow. As such, a list of tasks would be compiled into the backlog for developers to refer to. These tasks are divided up to the developer team(s).

These tasks are then worked on in parallel and independently between the developers. Once a task is completed, they will push it to the code repository to be integrated into the CI pipeline.

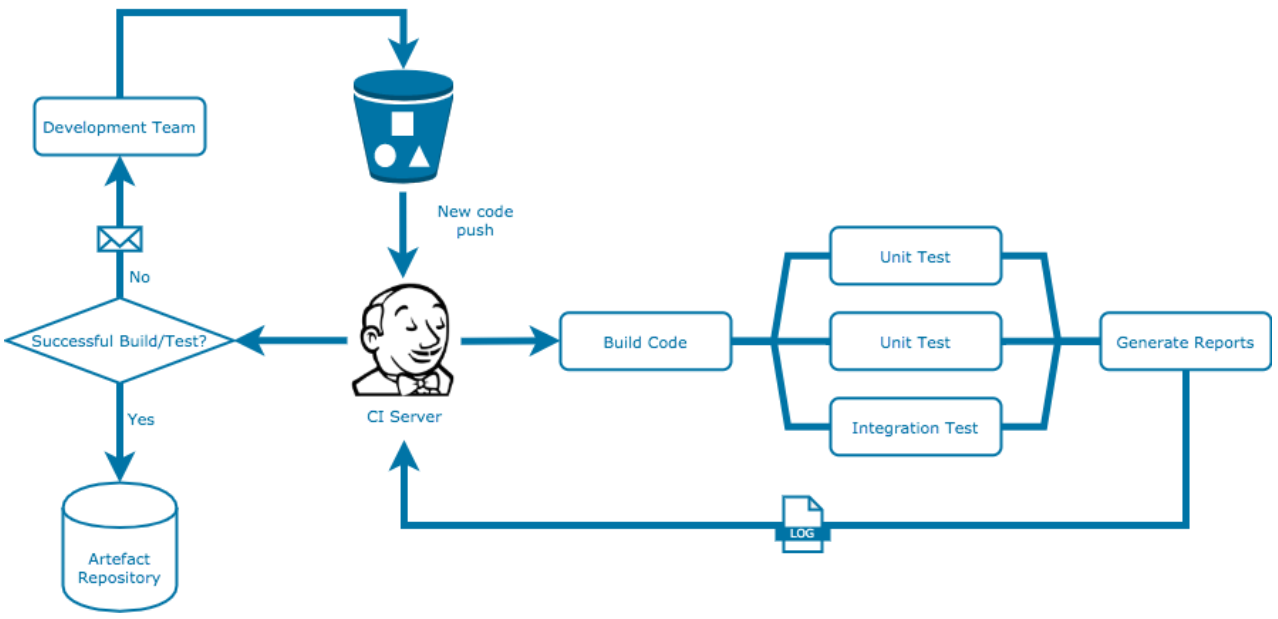


Code Repository

The code repository (often referred to as the *repo*) is where all the code is stored. It is managed using the VCS, which handles the integration of new code into the existing project. It checks for compatibility issues between the new and existing code and alerts the developer, making sure that a code push doesn't result in lost data.

Building and Testing

Once the push has been made, the VCS will trigger the **CI server**: a tool for automating integration and deployment processes. The CI server will build the application with the new code and perform acceptance tests to check the code works correctly.



If new code won't build or it fails any tests, integration is halted and the developer is informed of where the code failed along with detailed reports. This means bugs are caught long before they are integrated into the application and allows the developer to solve the problem quickly and efficiently.

These reports should contain such information as logs, error messages, build labels, build times, etc.

Successful builds that pass their acceptance tests get added to the **artefact repository**, ready to either be tested in a testing environment or sent straight to the staging/live environment to be served to the customer.

Benefits

There are a range of benefits to employing a CI pipeline to a production workflow that are not only limited to that of the developer and operations teams, but often positively affect the entire organisation.

This may be through cost-effectiveness, facilitating better planning, and greater transparency and understanding of underlying processes, allowing the company to better execute marketing strategies.

Some of the main benefits include:

Scaling

CI cuts out a large chunk of overhead occupied by manual code building/testing, slow communication channels both within and outside of the team, and highly stratified team structures.

Freeing up this time and energy in the production workflow frees up resources that can go towards scaling up the development team, code generation, code integration, and more.

Overly-planned releases schedules can be abandoned for more frequent updates, scaling team productivity.

Feedback Loop

Frequent and gradual feature updates allow for far more opportunities for business feedback. Teams can test new design ideas, new features, and get feedback about them faster, allowing for an agile approach to product

development.

It also allows for more client/customer feedback, as development teams can now show off new product features to the people they're being designed for, allowing them to adjust their product accordingly.

Incidentally, bugs and other issues can be rapidly fixed due to CI minimising the hassle needed for redeployment.

Communication

Leveraging VCS enhances communication between the teams as all changes are easily trackable. Teams and individuals are therefore more aware of each other's progress.

Stronger communication means teams/individuals avoid stepping on each other's toes and impeding each other's work.

Greater awareness of progress also aids transparency of work across the organisation. Other non-technical teams are much more able to review and understand what development teams are working on and how much they are achieving.

Challenges

While CI comes with a wide array of benefits, adopting the approach is not without its challenges.

Installation and adoption

The greatest hurdle for using CI is its initial adoption and technical configuration.

If there isn't an existing solution in place, the installation of the new pipeline is likely to be a long and involved process that has the potential to waste time, effort, and money should it be approached without enough planning.

Hence, before installing a CI pipeline there are considerations required about the existing engineering solution and how it will need to be built together.

A clear design approach with explicit and thoroughly thought out goals will facilitate a smoother adoption process with as little additional cost and effort required.

Learning curve

CI pipelines make use of many different and relatively new technologies that teams may not have any prior experience with, resulting in an initially high learning curve.

The main technologies required are version control systems, new hosting infrastructure, and container orchestration.

Not only is the technology likely to be different, but the workflow itself may also take some getting used to. For some members of the team, they may find their main responsibilities (e.g. testing) to have been largely automated, ultimately requiring them to adopt and adjust to a new set of responsibilities.

Tutorial

There is no tutorial for this module.

Exercises

Research different products that can be used at each stage of the CI Pipeline, to create your own CI Pipeline:

- Why have you chosen the product?
- How much does using this product cost?

- What are the alternatives to using this product?