Why CI/CD?

The short answer: Speed. The State of DevOps report found organizations that have "mastered" CI/CD deploy 208 times more often and have a lead time that is 106 times faster than the rest. While faster development is the most well-known benefit of CI/CD, a continuous integration and continuous delivery pipeline enables much more.

Development velocity

Ongoing feedback allows developers to commit smaller changes more often, versus waiting for one release.

Stability and reliability

Automated, continuous testing ensures that codebases remain stable and release-ready at any time.

Business growth

Freed up from manual tasks, organizations can focus resources on innovation, customer satisfaction, and paying down technical debt.

Building your CI/CD toolkit

Teams make CI/CD part of their development workflow with a combination of automated process, steps, and tools.

Version control

CI begins in shared repositories, where teams collaborate on code using version control systems (VCS) like Git. A VCS keeps track of code changes and makes them easy to revert if something breaks. It also enables configuration as code, which allows teams to manage testing, infrastructure, and more as versioned artifacts.

Version control stack

Shared coding environments, version control

Builds

CI build tools automatically package up files and components into release artifacts and run tests for quality, performance, and other requirements. After clearing required checks, CD tools send builds off to the operations team for further testing and staging.

Continuous integration stack

General automation, build tools, package managers, testing and code coverage tools

Reviews and approvals

Treating code review as a best practice improves code quality, encourages collaboration, and helps even the most experienced developers make better commits. In a CI/CD workflow, teams review and approve code in pull requests or leverage integrated development environments for pair programming.

Code review stack

Collaborative coding, code review tools, automated reminders, pull requests

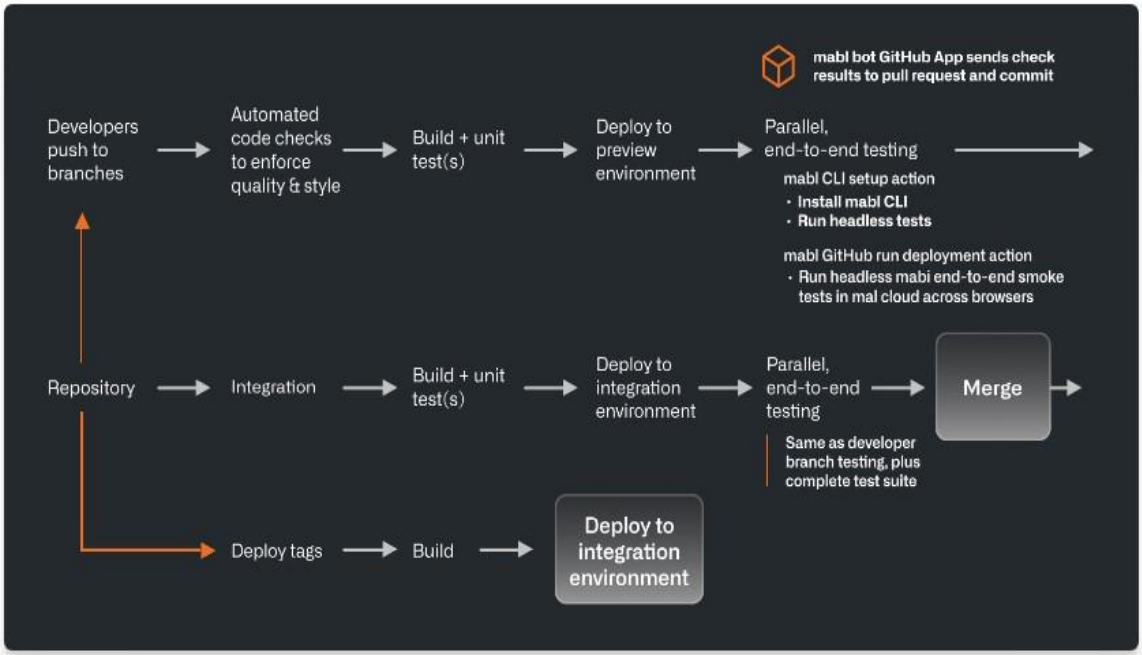
Environments

CI/CD tests and deploys code in environments, from where developers build code to where operations teams make applications publicly available. Environments often have their own specific variables and protection rules to meet security and compliance requirements.

Protected environments stack

Environments for testing, staging, and production, required reviewers and branch protection rules

Example CI/CD workflow



CI/CD doesn't have to be complicated, or mean adding a host of tools on top of your current workflow. At mabl, developers deploy to production about 80 times a week using only two CI/CD integrations: The mabl testing suite and GitHub Actions. Here's how it works.

1. Developers open pull requests to trigger initial builds and unit tests
2. Approved commits are deployed to a preview environment
3. Custom-built GitHub Actions install the mabl CLI and run headless tests
4. GitHub Apps provide live check results within pull requests
5. Approved commits are merged to the main branch for additional tests or deployed to production

What makes CI/CD successful

You'll find different tools and integrations everywhere you look, but effective CI/CD workflows all share the same markers of success.

Automation

CI/CD *can* be done manually—but that's not the goal. A good CI/CD workflow automates builds, testing, and deployment so you have more time for code, not more tasks to do.

Transparency

If a build fails, developers need to be able to quickly assess what went wrong and why. Logs, visual workflow builders, and deeply integrated tooling make it easier for developers to troubleshoot, understand complex workflows, and share their status with the larger team.

Speed

CI/CD contributes to your overall DevOps performance, particularly speed. DevOps experts gauge speed using two DORA metrics: Lead time for changes (how quickly commits are made to code in production) and deployment frequency (how often you commit code).

Resilience

When used with other approaches like test coverage, observability tooling, and feature flags, CI/CD makes software more resistant to errors. DORA measures this stability by tracking mean time to resolution (how quickly incidents are resolved) and change failure rate (the number of software rollbacks).

Security

Automation includes security. With DevSecOps gaining traction, a future-proof CI/CD pipeline has checks in place for code and permissions, and provides a virtual paper trail for auditing failures, security breaches, non-compliance events.

9. Reduce Costs

Automation in the CI/CD pipeline reduces the number of errors that can take place in the many repetitive steps of CI and CD. Doing so also frees up developer time that could be spent on product development as there aren't as many code changes to fix down the road if the error is caught quickly. Another thing to keep in mind: increasing code quality with automation also increases your ROI.

10. Easy Maintenance and Updates

Maintenance and updates are a crucial part of making a great product. However, it's important to note within a CI/CD process to perform maintenance during downtime periods, also known as the non-critical hour. Don't take the system down during peak traffic times to update code changes.

Upsetting customers is one part of the problem, but trying to update changes during this time could also increase deployment issues. Make sure the pipeline runs smoothly by incorporating when to make changes and releases. A great way to ensure maintenance doesn't affect the entire system is to create microservices in your code architecture so that only one area of the system is taken down at one time.

Conclusion

There are many tools that can help enable a smoother transition to a CI/CD process. Testing is a large part of that process because even if you are able to make your integrations and delivery faster, it would mean nothing if was done so without quality in mind. Also, the more steps of the CI/CD pipeline that can be automated, the faster quality releases can be accomplished.