

How do we know that we need CI/CD or Continuous Delivery?

There are several "warning signs" that teams exhibit that suggest they would be good candidates for CI/CD or Continuous Delivery. If you identify with any of these items, you should consider CI/CD an essential piece of your development workflow.

- Investing more time in a release cycle than delivering value
- Going through integration hell every time we finish a feature
- Code gets lost because of botched merges
- Unit test suite hasn't been green in ages
- Deployments contribute to schedule slip
- Friction between ops and development departments
- Only one engineer can deploy a system
- Deployments are not cause for celebration

What is CI?

Continuous Integration is the practice of automating the integration of code changes from multiple contributors into a single software project (definition from Atlassian).

CI Opens Doors

- Singular, Unified, Consistent Build Process
- Automatically Packaged and Downloadable "Binaries"
- Faster Recovery from botched Integration (If You Break It, You Fix It!)
- Unit Test Suite Gets Some Respect
- Greater Transparency and Communication
- More Time Adding Value (aka Developing Software)

An image showing many people with streams of ones and zeros connecting to a trophy which represents their shared goal.

Continuous integration is all about the source code.

New changes to the code need to be validated, verified, exercised, worked over, massaged and squeezed to see if there are leaks. We do this by compiling, transpiling, linting, running unit tests, performing static analysis, checking dependencies for security vulnerabilities and other things.

Continuous deployment is all about built code and deployment.

Once the source code has been built in CI, we're ready to ship it to servers and devices either in the same network or elsewhere. Depending on your team's delivery process and deployment strategy, you might deploy to a staging or pre-production server for final testing or you might deploy to production right away. Before doing so, CD can run scripts to prepare the infrastructure, run smoke tests, and handle rollbacks and reverts if something doesn't go as planned.