# ContinuousDelivery

*Martin Fowler*
*30 May 2013*

Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.

You're doing continuous delivery when: [1]

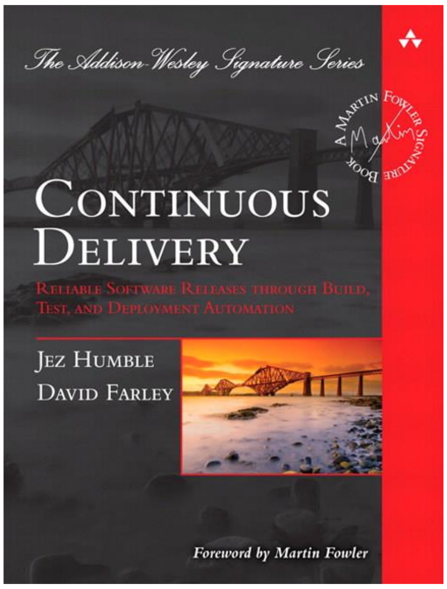
- Your software is deployable throughout its lifecycle
- Your team prioritizes keeping the software deployable over working on new features
- Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them
- You can perform push-button deployments of any version of the software to any environment on demand

You achieve continuous delivery by continuously integrating the software done by the development team, building executables, and running automated tests on those executables to detect problems. Furthermore you push the executables into increasingly production-like environments to ensure the software will work in production. To do this you use aDeploymentPipeline. The key test is that a business sponsor could request that **the current development version of the software can be deployed into production at a moment's notice** - and nobody would bat an eyelid, let alone panic.

To achieve continuous delivery you need:

- a close, collaborative working relationship between everyone involved in delivery (often referred to as a DevOpsCulture[2]).
- extensive automation of all possible parts of the delivery process, usually using a DeploymentPipeline

Continuous Delivery is sometimes confused with Continuous Deployment. **Continuous Deployment** means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day. Continuous Delivery just means that you are able to do frequent deployments but may choose not to do it, usually due to businesses preferring a slower rate of deployment. In order to do Continuous Deployment you must be doing Continuous Delivery.


The Addison Wesley Signature Series
A Martin Fowler Signature Book
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler

The book by Jez Humble and Dave Farley is the foundation book on this topic

Continuous Integration usually refers to integrating, building, and testing code within the development environment. Continuous Delivery builds on this, dealing with the final stages required for production deployment.

The principal benefits of continuous delivery are:

- **Reduced Deployment Risk:** since you are deploying smaller changes, there's less to go wrong and it's easier to fix should a problem appear.
- **Believable Progress:** many folks track progress by tracking work done. If "done" means "developers declare it to be done" that's much less believable than if it's deployed into a production (or production-like) environment.
- **User Feedback:** the biggest risk to any software effort is that you end up building something that isn't useful. The earlier and more frequently you get working software in front of real users, the quicker you get feedback to find out how valuable it really is (particularly if you use ObservedRequirements).

User feedback does require you to be doing continuous deployment. If you want that, but don't fancy getting new software to your entire user base, you can deploy to a subset of users. In a recent project of ours, a retailer deployed its new online system first to its employees, then to an invited set of premium customers, and finally to all customers.

## Further Reading

For more information the best online source is Jez Humble's Continuous Delivery page. (In particular Jez explains why he and Dave Farley chose the name Continuous Delivery and contrasts it with Continuous Deployment.) For more details, you should go to the book. I also list some resources on my guide page.

## Acknowledgements

Jez Humble provided detailed help with this page.

## Notes

**1:** These indicators were developed by the Continuous Delivery working group at ThoughtWorks

**2:** Despite the name "devops" this should extend beyond developers and operations to include testers, database teams, and anyone else needed to get software into production.

Updated on 12 August 2014 to add text on benefits and deploying to a subset of users.

## Translations: French