Integrate at least daily

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

By integrating regularly, you can detect errors quickly, and locate them more easily.

Solve problems quickly

Because you're integrating so frequently, there is significantly less back-tracking to discover where things went wrong, so you can spend more time building features.

Continuous Integration is cheap. Not integrating continuously is expensive. If you don't follow a continuous approach, you'll have longer periods between integrations. This makes it exponentially more difficult to find and fix problems. Such integration problems can easily knock a project off-schedule, or cause it to fail altogether.

Continuous Integration brings multiple benefits to your organization:

- Say goodbye to long and tense integrations
- Increase visibility enabling greater communication
- Catch issues early and nip them in the bud
- Spend less time debugging and more time adding features
- Build a solid foundation
- Stop waiting to find out if your code's going to work
- Reduce integration problems allowing you to deliver software more rapidly

More than a process

Continuous Integration is backed by several important principles and practices.

The practices

- Maintain a single source repository
- Automate the build
- Make your build self-testing
- Every commit should build on an integration machine
- Keep the build fast
- Test in a clone of the production environment
- Make it easy for anyone to get the latest executable version
- Everyone can see what's happening
- Automate deployment

How to do it

- Developers check out code into their private workspaces
- When done, commit the changes to the repository
- The CI server monitors the repository and checks out changes when they occur
- The CI server builds the system and runs unit and integration tests
- The CI server releases deployable artefacts for testing

- The CI server assigns a build label to the version of the code it just built
- The CI server informs the team of the successful build
- If the build or tests fail, the CI server alerts the team
- The team fixes the issue at the earliest opportunity
- Continue to continually integrate and test throughout the project

Team responsibilities

- Check in frequently
- Don't check in broken code
- Don't check in untested code
- Don't check in when the build is broken
- Don't go home after checking in until the system builds

Many teams develop rituals around these policies, meaning the teams effectively manage themselves, removing the need to enforce policies from on high.

Continuous Deployment

Continuous Deployment is closely related to Continuous Integration and refers to the release into production of software that passes the automated tests.

"Essentially, it is the practice of releasing every good build to users", explains Jez Humble, author of Continuous Delivery.

By adopting both Continuous Integration and Continuous Deployment, you not only reduce risks and catch bugs quickly, but also move rapidly to working software.

With low-risk releases, you can quickly adapt to business requirements and user needs. This allows for greater collaboration between ops and delivery, fueling real change in your organization, and turning your release process into a business advantage.

Our Continuous Integration product

We didn't just write the book on Continuous Integration. We also made the best build and deployment pipeline for on-premise and cloud in GoCD.

After many years of working with customers to construct deployment pipelines, ThoughtWorks developed Go as the first tool designed specifically for the practice of continuous delivery. It automates and streamlines the build-test-release cycle for worry-free, continuous delivery of your product. Go is now open source and free.

Download Now

Selected Insights

11 JAN 2016

Continuous Delivery

Architecting for Continuous Delivery

Vishal Naik

15 APR 2015

Software Testing

Enabling Continuous Delivery (CD) in Enterprises with Testing

Anand Bagmar

15 JUL 2014

Continuous Delivery

[Webinar] Continuous Delivery Into Practice With Go

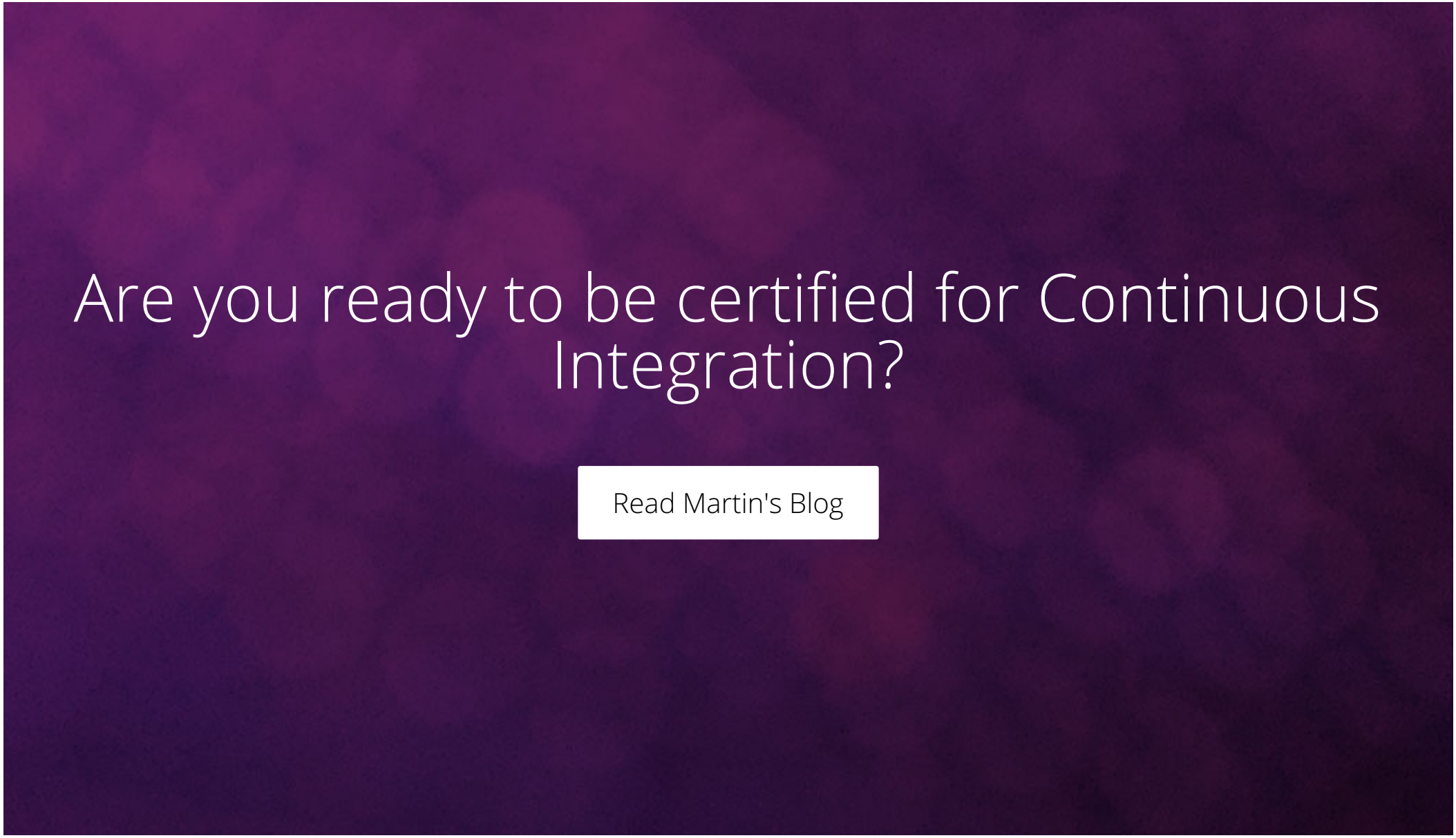
Go Team

Whitepapers and articles

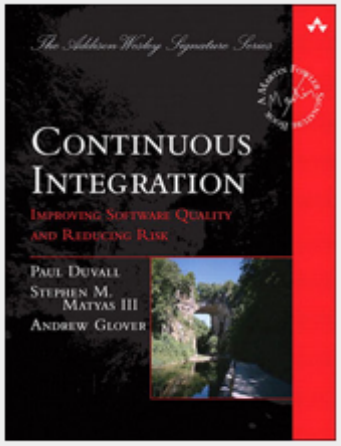
Continuous Integration - Martin Fowler

Continuous Delivery vs Continuous Deployment - Jez Humble

The Agile Maturity Model Applied to Building and Releasing Software - Jez Humble and Rolf Rusell

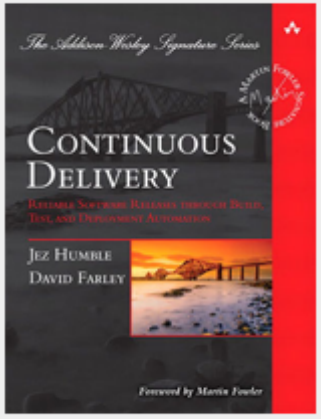


Read the books



Continuous Integration

Improving Software Quality and Reducing Risk by Paul M. Duvall



Continuous Delivery

Reliable Software Releases through Build, Test, and Deployment Automation by Jez Humble and David Farley

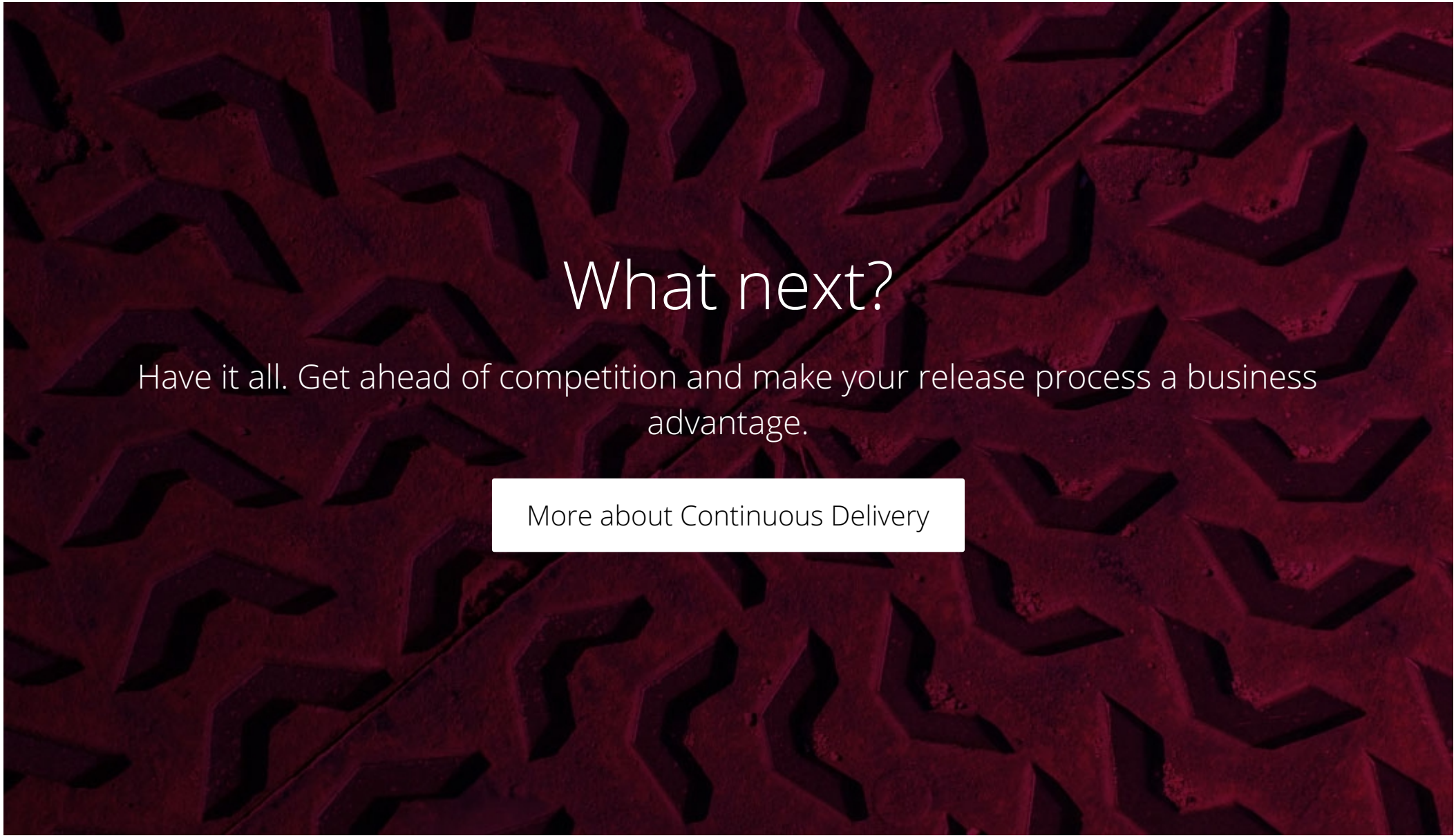
Watch these videos

View a one-hour summary of Continuous Delivery by Jez Humble and Martin Fowler recorded at YOW Melbourne.

Watch this video

Jez Humble interviews Michael Rembetsy, Director of Operations Engineering at Etsy, which manages to be PCI-DSS compliant while practicing continuous deployment.

Watch this video



About ThoughtWorks

We are a software company and a community of passionate, purpose-led individuals. We think disruptively to deliver technology to address our clients' toughest challenges, all while seeking to revolutionize the IT industry and create positive social change.

Connect with us