The **Tools** For Continuous Delivery

puppet labs

Table of Contents

Acknowledgements

Author: Aliza Earnshaw
Editor: Molly Niendorf
With thanks to Sze Wa Cheung, Christy McCreath, Gareth Rushgrove and Tim Zonca

Introduction

Organizations create, deliver and modify software to fill business needs. Those business needs are not static: They can change as suddenly as breaking news. Any organization using software to serve its customers — and really, that's almost *every* company, nonprofit or institution — has to find a way to develop, release and modify software more frequently and easily. That's why organizations of all sizes and types are implementing continuous delivery; they can't afford to be left behind by more responsive and agile competitors.

Martin Fowler's definition of continuous delivery sums it up well: Briefly, it's when any version of your software can be deployed to any of your environments on demand, and the team prioritizes keeping software deployable over releasing new features. Paradoxically, it gets a lot easier to release new features when you're working this way.

This paper offers an overview of the benefits of continuous delivery, the steps for launching it in your organization, and the kinds of tools you'll need to make delivery of small, frequent changes to working code an everyday reality.

Benefits of Continuous Delivery

When it's well executed, continuous delivery allows an organization to respond more quickly to its market and to customers, both internal and external. It also makes life saner for people in IT operations, software development and quality testing teams. Instead of long periods of development punctuated by looming deadlines, big dramatic releases and panicked remediation of serious bugs, software releases are small, predictable and less dramatic … even boring.

Deliver software with fewer bugs and lower risk.

- When you release smaller changes more frequently, you catch errors much earlier in the development process, because you are testing more often against existing code and infrastructure.
- Smaller changes are easier to roll back than huge, complex ones.
- Automated testing at every stage of development means you don't pass failed code to the next stage.
- Small changes make it easier to identify what element in new code broke something.

Release new features to market more frequently — and learn.

- When you release changes to customers more frequently — whether internal or external — you get earlier, more frequent feedback, telling you whether you're on the right track to pleasing the people who matter most.
- Enlisting your customers as development partners gives them a sense of co-ownership. As you create this pattern of collaboration and interaction, you build buy-in with internal customers and loyalty with external customers. And they'll be more ready to forgive you when you stumble (yes, sometimes that happens!).
- IT teams that provide their organization the ability to iterate and learn are providing a very high business value. The 2014 State of DevOps survey data showed that publicly traded companies with high-performing IT organizations had 50 percent higher market capitalization growth over three years than companies with lower-performing IT. Plus, these high-performing companies were more than twice as likely to outperform their own targets for productivity, profitability and market share than their peers with lower-performing IT.

Respond to market conditions more quickly.

- Market conditions change constantly. Regulations get modified; cybersecurity alerts are broadcast; consumer trends erupt like strange new diseases. (Your competitors aren't just standing still and watching, either.) Whether you've just discovered a new product is losing money, or that more customers are visiting your site from smartphones than laptops, it's much easier to make a fast change if you are already practicing continuous delivery.

Life is saner for everyone — IT operations, software development, QA, product owners, business line owners.

- Continuous delivery means the responsibility for software delivery is distributed much more widely — and shared responsibility makes life better. To achieve continuous delivery, you have to increase collaboration between teams all along the software delivery process, both by coming to agreement on goals and by aligning the work around a common toolchain. As your teams collaborate more effectively, power and responsibility can be shared even more. Best of all, a successful release becomes a shared success, one you can all celebrate together.
- Practiced well, continuous delivery can take a lot of stress out of software release. Big complex releases come with big risks, and are highly stressful. Even the smartest and most creative people can't come up with great solutions when they're exhausted from night after night of 2 AM calls, or weeks of 20-hour days fixing buggy code. Releasing smaller changes more often gets everyone used to a regular, predictable pace, leaving room for coming up with ideas and taking pleasure in the work.

Launching Continuous Delivery in Your Organization

Continuous delivery involves a number of tools and processes. In this section, we'll talk about the processes and practices you need to have in place, and in the next section, we'll highlight the tools you need.

Monitor everything.
Collect and analyze data on your production and test environments. You need to know where your baseline performance is, so you can see where to make improvements, and gauge whether you have actually made them. Monitoring tools also offer alerts and reporting.

Put all production artifacts into version control.
You need a version control system to keep a record of every version of every feature, add-on or other change to the code base. Infrastructure configurations and changes must also be checked into version control.

Developers must continually integrate new changes into the code base.
You need a continuous integration tool so you can integrate all changes, and trigger automated tests on every build. You also need reports to tell you which builds pass and which fail.

Automated tests.
These should be triggered by code check-in, so you don't pass faulty code to the next stage. Remember that you can build tests for many things: performance, scalability, security, internal policy violations, and more.

Automate the configuration of development, test, QA and production servers.
You need a single process for creating all environments so you can make production-like environments available early in the development process.

Code review.
Many organizations have change review boards. The 2014 State of DevOps survey found that organizations practicing peer code review were able to release code more quickly than those that required deployment approval from a change review board — *without* any increase in failed deployments.

Release smaller changes.

Bigger releases have more bugs, and they can be harder to sort out and fix, because there's so much more to check. CSG Systems International, an outsourced billing and customer care company, reduced release size by 50 percent and saw a steep decline in disruptions due to releases, and in disruptions in general. (If you're interested in learning more about how CSG applied DevOps principles in a large enterprise environment, you can watch a talk by Scott Prugh, CSG's chief architect and vice president of software development, at the DevOps Enterprise Summit.)

Institute blameless postmortems.

If you want continuous improvement (and you should), you need a culture that encourages and supports people when they surface bad news, and that views open discussion of failures as the path to improvement. Sam Eaton, director of engineering operations at Yelp, tells the story of how his ops team at Future PL moved from a culture of blame to a culture of learning together by notifying colleagues quickly about failures, and bringing "failcake." People appreciated being told up front what had happened, and the cake sweetened the deal.

Use dashboards to promote communication.

Many of the tools in the next section include dashboards. You want to use these for communicating the state of your infrastructure, number and types of incidents and other information to your colleagues, both in IT and outside.

The Tools for Continuous Delivery

Continuous delivery is a process that involves a number of tools. We've listed those that are most commonly used across a range of industries, in companies of all different sizes and stages of development.

Monitoring tools

- Graphite - An open source tool for storing data and rendering it graphically.
- Logstash - An open source tool for managing logs and other event data from your systems.
- Nagios - A monitoring and alerting tool for servers, switches, applications and services.
- Splunk - A tool for monitoring and visualizing data from your website, applications, servers, networks and other devices.

Version control tools

- Git - An open source version control system (VCS) that's distributed, so you can check in code and merge it while you are working offline.
- Mercurial - Like Git, Mercurial is an open source distributed VCS, so you can check in code and merge it while working offline.
- Perforce - A proprietary version control system that supports Git.
- Subversion - An open source version control system.
- Team Foundation Server - Microsoft's VCS. Includes continuous integration, issue tracking and product management capabilities.

Continuous integration tools

- Bamboo - A proprietary tool from Atlassian that runs builds and tests.
- Go - An open source continuous integration tool created by continuous delivery consulting firm ThoughtWorks, which offers paid support for Go.
- Hudson - An open source continuous integration tool with automated continuous build and monitoring of externally-run jobs (for example, cron jobs).
- Jenkins - An open source continuous integration tool with automated continuous build and monitoring of externally-run jobs (for example, cron jobs).
- Team City - Proprietary continuous integration system that integrates with Git and Mercurial.
- Travis - A proprietary continuous integration system.

Configuration management tool

- Puppet - Our configuration management platform is available both as an open source project and as a proprietary offering for enterprise companies. Puppet includes many other open-source projects, including Beaker for automated acceptance testing and R10K, an automated module deployment tool.

Code review tools

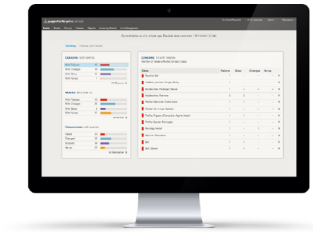
- Gerrit - A web-based code review system that enables online code review for projects using the Git version control system.
- GitHub - An online system for collaboration, code review and and code management. Available for free or as a for-pay enterprise service.
- Stash - A proprietary tool from Atlassian for reviewing code in Git, with enhanced security and other features for enterprise use.

Easier Releases and More Innovation

Continuous delivery reflects the fast-paced world of constant change that we all live in — and has helped to create that state. Releasing small changes more frequently gets your team into a different set of habits, turning what was once a stressful event into a regular, routine part of doing business. With continuous delivery, you get new ideas out to your customers more often, so you can learn from their responses and iterate further. It's how forward-thinking companies are doing business — and you should be, too.

Resources

- 2014 State of DevOps Report. The results of a survey of more than 9,000 IT professionals around the world, this report tells you the tools and practices that organizations are using to improve IT and business performance. Plenty of insight into organizational culture and stronger collaboration across technical departments, too.
- Automated Configuration Management: Why it Matters & How to Get Started. Without a configuration management solution, you'll end up managing packages, configuration files, firewall rules and other common settings manually. The risk of error is high, and you're consuming time you could use on projects and initiatives that deliver more value to your organization.



Puppet Enterprise is an IT automation solution that gives you the power to easily automate repetitive tasks, quickly deploy critical applications, and proactively manage infrastructure, both on-premise and in the cloud.

Questions?

Stop by Ask.PuppetLabs.com or contact our sales team.