Continuous Delivery: The First Steps

By Dave Jabs. Article as published in 

Article Reprint

AccuRev

MICRO FOCUS

Continuous Delivery: The First Steps

Continuous delivery integrates many practices that in their totality might seem daunting. But starting with a few basic steps brings immediate benefits. Here's how.

Software development groups that achieve high performance in development and delivery provide a strategic advantage to their business. However, many organizations struggle with delivering software in a timely manner. The set of practices called "continuous delivery" is gaining favor as an important part of the work of delivering new software on time. Continuous delivery defines a set of practices that aim to eliminate mechanical impediments and deliver software with greater velocity to respond to market needs.

Continuous Delivery as a Pipeline

- Let's start by outlining what continuous delivery is. One of my favorite definitions comes from Jez Humble of Thoughtworks, whose book is the seminal text on the discipline. He says, "The essence of my philosophy to software delivery is to build software so that it is always in a state where it could be put into production. We call this Continuous Delivery because we are continuously running a deployment pipeline that tests if this software is in a state to be delivered."
- Continuous delivery is not defined as continuous deployment, continuous release, or something that only cloud applications need worry about. Continuous delivery uses a set of key principles that involve staging the software development and delivery process. Each stage in the process possesses a distinct set of criteria that must be validated before software can progress to the next stage. These processes leverage automation for testing, software deployment, and promotion. The end result of the stage-based process is the formation of a delivery pipeline through which new software can continuously flow, and as it flows, it is validated to progressively higher levels of quality.
- Because continuous delivery is not a tool or product, achieving it is difficult. The approach to a successful continuous delivery process is a holistic one, where software development organizations must align technology, process, people, and values. It means a mindset shift for teams unused to it and, frequently, changes to processes and tools. For continuous delivery to be done properly, it also requires organizational changes. Achieving excellence in continuous delivery isn't easy, but the reward is enormous: Development teams will be able to move at velocities not previously thought possible.

Steps to Continuous Delivery

The overarching concept of continuous delivery is not new; in fact, the first Agile principle states, "the highest priority is to satisfy the customer through early and continuous delivery of valuable software." Despite this familiarity, many companies—even those committed to Agile processes—struggle to just get started.

The roadmap to achieve continuous delivery begins with modeling the development and delivery pipeline. Many of today's organizations are formed from disconnected teams. If you want continuous delivery, you need continuous flow from beginning to end.

Achieving continuous flow requires a multi-step approach whereby organizations:

1. **Model and measure the cycle time in your existing processes**
2. **Identify delays in the process**
3. **Develop action plans to minimize current delays and eliminate potential future delays**

Following this process allows organizations to break down the silos between development and operations in order to produce a single, unified pipeline ranging from concept all the way through delivered software.

Continuous delivery is a journey, not something that is achievable overnight, because development organizations must make significant changes, often to technology, process, culture, and people, to implement the full pipeline — and these changes take time, often more than a year.

Where should software development organizations start? Here are three fundamental areas where companies should focus their initial efforts.

1. **Requirements management.** Requirements must be decomposed to minimum deliverable units; development plans should be created and organized around incremental delivery.
2. **Testing/Validation.** The QA process has to be empirical, structured and organized, as well as automatable. In addition, good continuous integration is a prerequisite for continuous deployment
3. **Delivery Mechanics.** Start with version control. Reproducibility of software builds is a common practice, however reproducibility of deployment is often a missing element. Then, automate delivery. Keep at it until it's a oneclick frictionless process.

Barriers to Successful Continuous Delivery

Most software development organizations face what appear to be overwhelming barriers to achieving continuous delivery. They find themselves in this perceived situation because their existing legacy products have been developed without following continuous delivery principles. The good news: They can start now. Companies can begin defining their goals and objectives and put practices in place that allow them to make small gains along the way to their ideal situation.

Continuous delivery integrates many practices that in their totality might seem daunting. But starting with a few basic steps brings immediate benefits.

Here are a few common areas where companies seem to have the biggest struggles with regard to implementing effective continuous delivery strategies.

Requirements Management

Companies often find themselves struggling to define small deliverable stories that are easily understood, managed, and delivered. The solution to this problem is to set goals to make each story smaller, and then focus on continually improving the story writing process until stories can be finished with very short cycle times. Feature toggles can be used to hide stories until a complete set of related stories are completed, or to allow A/B testing of the usability of a set of stories.

Focusing on the Wrong Tests

The state of software development testing today sees most organizations investing heavily in GUI testing. This practice is actually an impediment. Organizations should put the majority of test focus on unit testing, where test coverage is at its greatest. From there, focus should next be placed on integration testing, in which a small number of integration tests are used to ensure all the pieces work together. GUI testing should have the least amount of focus because GUI tests are often brittle, slow, unreliable, and the most expensive to maintain for the value they deliver. The result of this testing strategy will yield faster cycle time, higher quality, and lower test maintenance costs.

Scaling Continuous Integration

Another frequent problem is scaling continuous integration. As development scales, you get many more concurrent changes in progress, builds take longer and tests take longer, resulting in broken builds. In the worst case, builds on a single mainline become mostly broken instead of mostly working. That is not continuous delivery. Worse yet, the unstable main line affects all developers and slows down their productivity dramatically.

There are two primary techniques companies employ to achieve continuous delivery at scale. Each of these techniques is designed as a "divide and conquer" strategy, allowing for focus on specific areas which can have significant impact on continuous delivery:

- **Component or Service Architecture.** Companies can choose to change their software to a component or service architecture. These are components that can be individually deployed in production independently of other components. They have clean interfaces and are compatible across multiple releases of components. This structure provides scalability because you can then run separate development and delivery pipelines that don't interfere with each other.
- **Multi-Team Continuous Integration.** In this approach, scalability is achieved by doing development using separate teams. You then join the work from those teams as they go through integration and test processes, and eventually, out to the deployment pipeline. A key to multi-stage continuous integration is to break down the development pipeline and integration testing into stages. Organizations may define these stages by team branch, feature branch, build (nightly, weekly, etc.), and main line. Using integration testing along each of these stages provides a level of test coverage and continuity that allows code to flow up the pipeline while also identifying issues faster and closer to the developer, where the cost to fix the issue is less expensive.

Embodying Continuous Delivery

An example of successful scaling continuous delivery I have seen is with an international online gaming company with 70 developers organized into 10 teams. One of the keys to their success was the change management process they have incorporated into their continuous integration practices. Due to the nature of the gaming business, inter-product dependencies cause a rate of change per product on a daily basis that is sometimes as high as 60%. Using effective continuous integration and change management techniques, this company was able to handle this rate of change, and as a result, is producing more than 3500 releasable builds per month. This company fully embodies the continuous delivery principle of always having releasable code.

Conclusion

Micro Focus can help your organization to use continuous integration and continuous delivery effectively, which dramatically improves the way your engineering team operates, as well as the business overall. Following these practices will provide significant competitive advantages in software development time-to-market, agility, and quality.

To find out how Micro Focus® AccuRev can help your company to achieve successful Continuous Delivery, go to: **www.borland.com/AccuRev**

Micro Focus
UK Headquarters
United Kingdom
+44 (0) 1635 565200

U.S. Headquarters
Rockville, Maryland
301 838 5000
877 772 4450

Additional contact information and office locations:
www.microfocus.com
www.borland.com