LYONSCG
eCommerce Realized!

Innovation & DevOps Culture

By Stephen Chinn, July 2016

Staying ahead of your competition requires continuous innovation. A DevOps culture is an essential foundation for any eCommerce team that needs to reliably and efficiently deliver innovative store features.

DevOps refers to a working approach that melds together the perspectives of Development (i.e. writing software) and Operations (i.e. running systems). Traditionally, these have been separate functions, managed by different groups, but the need for shorter and faster cycles of feature development and the need for nimble management of cloud infrastructure have made it essential that these roles work in lockstep.

FASTER DEVELOPMENT

A successful eCommerce business must be fast, reliable and innovative. To retain customers, on-line retailers must provide all the features customers have come to expect. It's a list that keeps getting longer. To attract new customers, retailers must find ways to offer compelling new features that customers can't find elsewhere. Of course, if a new feature is successfully attracting new customers, it will only be a matter of time before it gets copied. Soon, it will be just one more item on the list of features everyone expects to have and the next new feature will be required to keep your competitive edge.

Getting the most out of a new innovation, or if you're not currently in the lead, catching up before too many customers are lured away, makes the time-to-market of feature development exceptionally important.

Time to market (TTM) is the length of time between a product being conceived of and it being available for sale. In the world of eCommerce software development, TTM is the time it takes to go from conceiving of a new feature to getting that feature successfully working for your customers. There's great value in having an appealing feature that your competitors don't have, but not every new feature will be "a hit" with customers. Keeping up or catching up; either way, there's a lot of pressure to keep developing new features as quickly as possible.

The desire to adapt and develop applications quickly has been the driving force behind the wide spread adoption of Agile development processes. First introduced in 2001, Agile has become the default industry standard for business-oriented software development. Agile is based on 12 principles. The first three directly reflect the importance of TTM.

1. *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2. *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3. *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*

These are the goals of an Agile team. The appeal of these principles has not been lost on the business community. From there, the principles go on to address how Agile processes will achieve these goals.

4. *Business people and developers must work together daily throughout the project.*
5. *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
6. *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7. *Working software is the primary measure of progress.*
8. *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9. *Continuous attention to technical excellence and good design enhances agility.*
10. *Simplicity--the art of maximizing the amount of work not done--is essential.*
11. *The best architectures, requirements, and designs emerge from self-organizing teams.*
12. *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*

Principles behind the Agile Manifesto (2001), http://agilemanifesto.org/principles.html

Though customers invariably, and for good reason, want the benefits promised in the first three principles, they are often not prepared to facilitate a working process that follows the remaining nine. As consultants it's our task to show them the way.

Though a good on-line store makes purchasing simple, beneath the surface, an eCommerce platform is a complex mechanism that integrates into mission critical business systems and connects with a wide array of supporting services. Making significant changes to it requires input from a team of people who must cover a wide variety of skills and interests. Delivering error free

features, both quickly and efficiently, requires a team that's extremely well organized and practiced at working together. The team needs to be well-balanced before it can ramp up to a rapid and sustainable productive cadence.

Since 2001, the Agile way, once a revolutionary force, has become the established orthodoxy and eCommerce, now an essential part of almost every business, has transformed customer expectations of software and the rate at which software evolves. There has also been a revolution in the way the infrastructure is managed.

FLEXIBLE INFRASTRUCTURE

"The Cloud" has brought with it the promise of infinite adaptability and endless resources. It has led some to think that infrastructure management is a less important concern than it was, but the reality is, ignoring infrastructure is, at best, extremely costly. At worst, poor infrastructure management can completely undermine your on-line business.

Cloud systems, like Amazon Web Service (AWS) or Azure (Microsoft's Cloud Service), are designed to distribute processing across a large pool of resources in ways that promote the full utilization of those resources. The problem with the traditional computing model, in which businesses use dedicated physical servers to run their applications, is that, even in well-organized systems, most servers are idle 80% of the time. Cloud systems hope to solve this problem by providing a large pool of servers to a very large pool of users.

In theory, if the pool of users is big enough (e.g. encompassing a huge number of businesses, distributed all around the globe), just random variation in computing need would result in resource utilization rates considerably higher than 20%. In their *2014 Data Center Efficiency Assessment* report, the Natural Resources Defense Council estimated cloud server utilization at 65% compared to on premise (i.e. companies running their own in-house physical servers) utilization at 12 to 18%. Utilization of co-located managed physical servers is typically estimated at around 30%.

Yet, despite the hopeful predictions of cloud evangelists, so far, it's not been massive user pools and random variations that have kept cloud utilization rates high. It's been the harsh penalties companies have experienced when they've jumped into the cloud with the expectation that it will automatically help them run efficiently.

The cost of running a cloud server in a 24/7 mode, at a fixed level of power, is extremely high, and the bigger the server gets, the more extreme the cost gets. If you tie up a whole physical server, a cloud provider is going to make you pay. By forcing that server to be dedicated to you alone, you have eliminated the utilization advantage cloud providers rely on to run profitably. It's the ability to run at significantly higher utilization rates that creates the opportunity for cloud customers to pay less for computing power, and for cloud providers to generate healthy profits from their service. If you really need a lot of server power in a fixed configuration, you need a physical server and should use a hybrid architecture. Hybrid architectures use a mix of physical and virtual (i.e. cloud) servers.

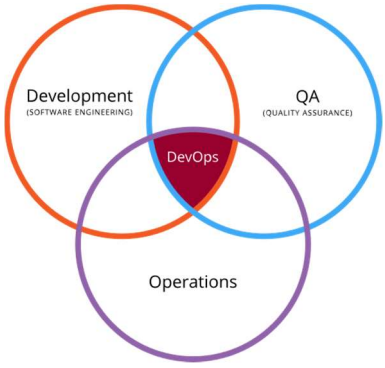
To make the cloud pay, cloud providers have developed a wide variety of tools that can dynamically change the size and number of servers used by your eCommerce platform. These tools allow cloud customers to create, scale, and destroy servers programmatically. For example, every night your company needs to run a set of analytical processes on sales data collected through the day. To do this efficiently in the cloud, you would program the cloud environment to create a large server at 11:30 pm, run the analytical process between 11:30 pm and 12:30 am, and then destroy the server at 12:30 am. The cost of running the cloud server for just 1 hour a day, though much higher per hour than a similarly capable physical server, is less than paying for the physical server for 24 hours a day.

The potential savings are significant, but they are all predicated on knowing just how much power you are going to need and programming the cloud environment to provide just the right amount of power at just the right time. This layer of infrastructure programming is as important to the successful introduction of new features as is the programming of the feature itself.

DEVOPS CULTURE

Wikipedia defines DevOps in the following way:

***DevOps** is a **culture**, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.*



DevOps is the evolutionary and necessary response to the challenge of managing continuous feature optimization in a world of programmable infrastructure. The emphasis is on close collaboration between program managers, software

developers, infrastructure administrators, and quality assurance engineers, and the automation of processes to make infrastructure environments highly efficient and adaptive.

For production environments, building elasticity is key. An elastic system can expand to meet growing need and shrink back to its smaller “normal” size when the need has subsided. Cloud environments provide the possibility of elasticity, but it takes thoughtful design and programmed automation to realize that possibility.

As important as production environments are, it’s in the supporting development and test environments that the true hallmarks of a DevOps culture can be found. By automating the creation and destruction of these environments, the cost of creating environments that faithfully reproduce production-like conditions becomes affordable. As development efforts ebb and flow, supporting environments adapt to the needs of the team. Multiple development environments can be quickly constructed to support multiple overlapping sprints. When they are no longer needed these environments are shut-down or destroyed.

Automated testing processes ensure that a complete set of tests can be efficiently run before any changes are pushed into production. Load testing systems spin up to simulate high volume traffic and spin back down again after their testing cycle is complete. And the combination of well-designed performance tests and faithful production representations allow the DevOps team to predict the performance impact of the new features.

By comparing the cost of improving code efficiency against the cost of providing extra infrastructure power, the team can pick the most cost efficient way of providing a new functionality. A new feature can be tried using an optimized investment in development effort. If it is successful, production infrastructure elasticity will ensure reliability as traffic grows, and further investment in code efficiency can be made when there is no doubt about the value of that investment.

A DevOps culture, and all that it entails, drives successful innovation by facilitating the continuous development of new features in a rapid, reliable, and cost effective way. In today’s fast paced eCommerce market place, if you want to stay ahead of the competition, developing a DevOps culture gives you a crucial advantage.

DEVELOPING A DEVOPS CULTURE

Developing a DevOps culture requires a long term perspective. It takes investment and patience. This might appear to be at odds with the purpose of a DevOps culture, but it's the long run that really counts. Picture Henry Ford tinkering with his assembly line, while people in the background grumble about how such foolishness just gets in the way of making cars.

A century later the value of automation is unmistakably and certainly automation is a crucial part of a DevOps, but automation tools are not the key to a DevOps culture. Automation eliminates "busy work" so that people can devote their time to tasks that require flexibility, creativity, and nuance, all things machines, even computers, don't do well. The core feature of a DevOps culture is the high performance, cross-functional, team. Automation is used to concentrate the team's productivity and facilitate its cooperation. Teamwork is the key to DevOps.

TEAMS

Stressing the value of teamwork is nothing new. Andrew Carnegie called teamwork, *"the fuel that allows common people to attain uncommon results."* But emphasis on the importance of teams and the dynamics of good teamwork, has grown steadily as the complexity and pace of work has increased. The members of a DevOps team must perform well as a team, so it is worth reviewing what that entails.,

In their 1993 book, *The Wisdom of Teams*, McKinsey & Company partners, Jon Katzebach and Douglas Smith defined a team as, *"a small number of people with complementary skills who are committed to a common purpose, set of performance goals, and approach for which they hold themselves mutually accountable."* Their conclusions about what makes a high performance team provide some insight into why creating a high performance DevOps team can be difficult. They wrote: *Behind high-performance teams lies a story of commitment… a high-performance team must have a small number of people with the required skills, purpose, goals, approach, and accountability… what sets apart high-performance teams, however, is the degree of commitment, particularly how deeply committed the members are to one another.*

A good DevOps team must encompass many perspectives and a wide range of skills. In the past, when development cycles stretched over months, a project manager could let people with distinct perspectives and specific skills, contribute to the development process by shepherding it through various silos of competency; marketing, engineering, operations, etc. Today's Time-to-Market

pressures have made this old approach impractical. Yet, it still takes all of these competencies to deliver a quality product. If there's a gap in skills, the team will make costly mistakes, but large teams tend to get unwieldy and slow processes down. Getting the team just right requires striking a careful balance. Member contributions must be carefully measured and applied in short rapid bursts, it takes discipline.

Encompassing all the required skills, often involves creating teams of people who cross, not only the traditional divisions of profession, but who work for different companies, and live in different cities. This doesn't precluded the formation of a high performance team but it does imposes additional challenges. It takes time for such disparate people to form the bonds that lead to productive collaboration, shared vision and team commitment.

High performance teams take collective responsibility for their performance. For the individual members of a high performance team, the most immediately poignant source of accountability is the team itself. This type of accountability is powerful because it effects every interaction within the team. A cohesive team is essential and investments that increase team cohesiveness, like getting everyone together in the same room with some regularity, are essential for the creation of an effective DevOps culture.

A singular focus on increased speed and an over-emphasis on individual, rather than team, accountability, are merely attempts at speeding up the movement of projects through the old ridged silos. A lot of frenetic energy is created but actual productivity is likely to go down. Real productivity gains occur when a well-rehearsed team expertly juggles multiple interlocking cycles of development. It might, at times, look frenzied, but it is not chaotic.

There is a cadence to the work of a high performing DevOps team. Don't mess with it lightly. Velocity is achieved through the expert and disciplined execution of carefully crafted processes. The team and its processes are designed to be flexible and adaptive but they are not unbreakable. The team's balance has to be maintained while its pace gains momentum.

A DevOps team isn't a substitute for a good planning, indeed, it's wasted without it. Constantly shifting priorities, unrealistic expectations, and poor planning, will quickly bring the pace of development work to a crawl. Once the team is running in high gear, respect its processes, keep feeding it a steady flow of work, and be ready to innovate.

TOOLS

There is no ultimate set of DevOps tools that, once installed, will transform a development team into a DevOps team. A DevOps team invests in the automation of a very specify factory. It's the factory that's used to make and continuously remake your ecommerce system.

DevOps requires a shift in emphasis. If you want to build cars faster, you can't do it by focusing solely on the cars. You have to look at the factory. Look at the assembly lines and the quality of the machines on them. How efficient are they? How flexible are they? Only by investing in a first class factory do you make it possible to build first class cars.

Quality control is supremely important. Speed acquired by skipping over quality control is just brinkmanship; in the long run you will see only short bursts of speed that are soon wholly negated by resulting problems. The consequences get even worse if you misinterpret what happened. You may end up rewarding the behavior that created the problem and punishing the behavior that fixed it.

The conditions that allow high velocity development are maintained by thinking long term. DevOps requires ongoing investment in automated testing. There is a symbiotic relationship between the production system (e.g. the e-commerce system) and the testing system. Thinking about how to test a new feature should be occurring in parallel with thinking about how the new feature will work. Tests should be automated. The evolution of the system should involve moving two sets of code, testing and tested, down parallel tracks.

Infrastructure elasticity and programmability have been a boon for DevOps. Because environments don't have to be on all the time, team can afford to build non-production environments that are more sophisticated and comprehensive than ever before.

Performance testing in particular improves when testing environments can be more faithful representations of the production environments they simulate. If a service or functionality runs on an independent server in production, it should be tested on an independent server. This way its performance profile will match its production equivalent, in form at least, if not always in scale, and its reaction to load (i.e. high site traffic) will provide a more accurate picture of how the production environment will behave under similar load. In the past, the cost of maintaining such faithful simulations was often deemed cost prohibitive but now cloud technology and automation

make it possible to build environments efficiently and to simply turn them off when they aren't being used.

Temporary scaling of servers also helps with the creation of better testing systems. Because powerful testing servers cost less to maintain, they can be used regularly and integrated into the normal course of a sprint (i.e. a cycle of development). Load generating servers in the cloud can take on tremendous processing horsepower for the time it takes to run a test, then shrink to a tiny holding size, or simply turn off, when the test is over.

Truly rapid development requires the management of multiple simultaneous sprints. Infrastructure automation makes it possible to build and destroy multiple specialized development environments that feed into single a pre-production environment. The number of parallel development environments can expand and contract with the ebb and flow of development work. By leveraging the flexibility of programmable infrastructure a DevOps team can efficiently maintain environments that facilitate their highest possible level of productivity.

TIME

There is an interesting parallel between the way you maximize the value of infrastructure and the way you maximize the productivity of a DevOps team, at the root of both is time. The question that has to be asked is, when something should occur. Most physical servers are largely idle most of the time and then, when they are needed, they can't move fast enough. We solve this problem one of two ways, we make the server bigger or, we set the server working before the moment of need, preparing all that it can, so that when the need arises, it can perform as efficiently as possible. The second approach requires more thought and planning but it is clearly better. By moving when things happen and by investing ahead of the critical moment, we have increase the productivity of the server without having to increase its power.

A DevOps culture increases development velocity by increasing team productivity. It is as much about when things are done as it is about what things are done. Investments are made, so that at critical moments, the team can move with stunning swiftness, not because they are suddenly more powerful but because they are extremely well prepared. They haven't suddenly increased their speed, far from it, they maintain a steady methodic pace. What is witnessed in those moments is the concentrated impact of work done long before the moment. The team moves fast because they are practiced at working together. They move through processes quickly, without skipping steps, because they've prepared everything ahead of time.

CONCLUSION

DevOps is not a magic formula that increases the number of hours in the day. It will not increase the speed of typing or reading or pouring coffee. It won't eliminate the need for great ideas to germinate in the minds of those that have them. Least of all, is it a new role for an individual person, it shouldn't be a job title.

A DevOps culture is created by building carefully balanced, cross-functional, high performance teams and equipping those teams with tools that, they will use to build specialized factories; factories designed to efficiently create and recreates your e-commerce systems. The result is a capacity to bring innovative ideas to your customer at a rapid rate. Creating a DevOps culture requires investment but in market place driven by continuous innovation a DevOps culture is the essential foundation of success.