DevOps: Adopting and Adapting to a Culture of Continuous Improvement

- Introduction ................................................................................................................................................ 2
- What is DevOps, How Did it Develop, and Where Does it Go from Here? ........................ 4
- Building a DevOps Culture .................................................................................................................... 11
- Implementing DevOps: Adopting and Adapting to a Culture of Collaboration ............... 24
- Measuring DevOps: Proving ROI on Your Investment ............................................................... 46
- Getting Started Building Your DevOps Initiative ......................................................................... 52
Introduction

Far more than just a methodology, DevOps has been credited with enabling organizations of all kinds all over the world to achieve better results, faster, and less expensively, a trio of value formerly considered impossible to achieve!

Because it is still relatively new there are still many misconceptions about exactly what DevOps really is:

» It can be characterized as a process in that it contains a wide variety of processes within it.

» It can be seen as a strategy in that it brings multiple teams together to achieve common goals.

» It can be considered a methodology, in that it generally defines a series of steps that will be executed to most efficiently achieve various objectives.

» It can be part of any organization’s overall digital transformation as it enables them to improve their ability to further integrate more impactful software.

» It can be called a component of your company’s culture in that it is pervasive well beyond the two departments that comprise its name, development and operations.

DevOps drives continuous integration to enable constant delivery that will consistently result in constant improvement.

Constant Improvement

In this eBook you’ll learn how DevOps completely transforms software development in ways that may, at first, seem counterintuitive. For example, instead of making users wait months for their next software upgrade DevOps enables much faster delivery of far smaller, more modular updates each created and maintained by dedicated teams, with some DevOps organizations executing dozens of software improvements per day.
By partnering with operations who return user feedback faster than ever, developers quickly respond with more new improvements. This is an iterative process of develop-deploy-use-feedback-repeat. Each iteration improves performance thus achieving the sought-after state of constant improvement.

A Foundation for Adoption

Organizations need to carefully consider what their return on the investment of adopting DevOps will be before they can make that decision. A key goal of this eBook is to provide effective context and key information required to support that decision, including:

- A discussion of the definitions and derivation of DevOps.
- DevOps development as an outgrowth of Agile, Lean, and other methodologies.
- Goals and objectives that would benefit from DevOps Implementation.
- The steps required for implementation of DevOps in your organization.
- How to measure your DevOps results.
- Effective ways to get started with your DevOps initiative.

At the end you’ll be prepared to meaningfully evaluate DevOps usefulness for your organization, and to take appropriate next steps. The most important questions to answer for those considering the adoption of DevOps are whether or not you have the desire for superior performance and the tenacity to see the program through.
What is DevOps, How Did it Develop, and Where Does it Go from Here?

The simplest and most understandable definition of DevOps can be drawn from the two words of which it is a compound, development and operations. As we move along you may find it useful to change the question “What is DevOps” to “What can we achieve by implementing a DevOps culture?”

In the earliest days of computer programming, a programmer produced the code necessary to have the computer perform specific functions. That code was presented to the computer operations team who loaded it and ran it. If something went wrong, those computer operators really had no idea how to fix it, so they returned the code to the developers to make corrections. As you can imagine, this could become very time-consuming very quickly, and did.

DevOps celebrates the value of collaboration, communication, and integration, making them the three main principles of this ever-growing, modern approach to software delivery. In this case, it’s collaboration between the developers of the software and the operators of the hardware. They come to share methodologies and tools working as a single united team.

Before There Was DevOps, There Was Agile

We usually find ourselves using the word “agile” to describe athletes, dancers, and other people who move quickly, gracefully, and with precision. Those qualities led to the adoption of the word to describe a new methodology that began a radical change in the way software gets developed.

When “DevOps” was first developed in 2009 and named by Patrick DeBois, he described it as an extension of “agile” development environments that aim to enhance the process of software delivery as a whole.
In this context, agile refers to the Agile Software Development model that was developed to improve upon the classic “waterfall” method in which each step cascades from the previous step. This was far too slow to accommodate ever increasing needs for speed in development and deployment of software. DeBois and his team developed an “Agile Manifesto” with a fairly simple and straightforward set of values and principles:

The Agile Manifesto’s values are:

» Individuals and interactions over processes and tools.
» Working software over comprehensive documentation.
» Customer collaboration over contract negotiation.
» Responding to change over following a plan.

The developers of the manifesto point out that there is value in the items on the right side of their list, but they value the left side items more.

Agile’s principles are:

» Highest Priority is Customer Satisfaction
» Progress Measured by Working Software
» Welcome Changing Requirements
» Sustainable Development Pace
» Frequent Delivery of Software
» Continuous Attention to Technical Excellence
» Businesspeople & Developers Cooperating Daily
» Simplicity
» Build Projects Around Motivated People
» Self-Organizing Teams
» Face-to-Face Conversation is Best
» Regular Reflection & Adaptation
While you’d like to think otherwise, the fact is that many of these principles are a radical departure from what normally happens in the IT departments at most organizations.

» The highest priority for the software development team may be the completion of more items from their “stack.” Operations may focus on avoiding complaining users. The customer often doesn’t show up in the discussions.

» The status quo represents “peace and quiet” but not progress. Change is, at best, unwelcome.

» Updates and new versions, when they do show up, are downright feared, with the result of many users staying with the old version and creating all manner of havoc.

The emphasis is on process and product rather than people. By contrast, the Agile methodology talks about individuals and their interactions, collaboration with each other and with customers. Reflection before reaction.

**DevOps Takes Agile to the Next Level**

DevOps added a few new principles, most especially the concept of CI/CD (continuous integration and continuous delivery). This spoke most directly to the emerging environment everyone found themselves living in. Everything happened faster, so businesses needed to respond faster. Time was of the essence, so there was a need to find a way to constantly be improving upon your software to help you remain competitive in a highly dynamic world.

Therein lies the promise of DevOps, an environment of constant, interactive improvement to the way people get their work done.
The Evolution of Lean to Agile to DevOps

Many try to compare and contrast Lean, Agile, and DevOps methodologies, but the reality is that Lean led to Agile which was then extended to become DevOps. The ultimate goals of all three are similar, including faster time-to-market, faster time-to-value, greater efficiency, greater quality, increased customer satisfaction, time and financial savings, and constant improvement through constant delivery and rapid user feedback. While Agile and DevOps are usually considered to be focused on software development, Lean thinking began and continues to benefit many industries. The continuity between all three provides an ever-evolving set of definitions that is, in and of itself, in a state of constant improvement.

The Problem with the Personal Computer Revolution

At around the same time the personal computer was becoming ubiquitous. At that time, developers estimated that it took about three years to produce a new application, a challenge they referred to as “application delivery lag.” Business was moving far faster than that and accelerating at a rapid pace.

In 2001, a group of 17 developers, all Lean practitioners and advocates of then-existing software development methodologies such as Scrum, extreme programming, crystal, adaptive software development, feature-driven development and dynamic systems development, gathered to discuss the creation of more lightweight frameworks to help accelerate software development processes significantly.

They named their creation “agile”, developed and wrote a “Manifesto for Agile Software Development” and created a non-profit organization called the Agile Alliance. Today there are over 30,000 members and subscribers participating in what is commonly referred to as the Agile movement.

Lean led to Agile.

Some prefer to refer to Agile as a way of thinking or a philosophy since it encompasses so many methodologies, such as Scrum, Extreme Programming, Kanban, pair programming, and others, and is not itself a methodology.
The Rise of DevOps

By 2007 the process optimizing of Lean manufacturing with its ultimate goal of continuous improvement, and its offspring, Agile, was replacing the traditional “waterfall” methodologies of software development that had been too slow with rapid, iterative methods that increased the speed of software delivery tremendously.

This was also the time of the early maturation of cloud computing including infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS) with incredible implications for integration and software-driven-everything. Up until then the developers who created the code and the operators who managed the infrastructure had a difficult relationship that often culminated in mutual blame whenever something failed. Perpetual finger-pointing.

By 2009 as infrastructure was, itself, moving toward being driven by code, it became more and more obvious that developers and operators needed to interoperate seamlessly, perhaps even using the same tools and platforms to assure a contiguous path from development to delivery and undergoing intensive cross-team training, all with the common goal of creating an environment of constant improvement.

A 2013 Gartner report explained that “DevOps movement was born of the need to improve the agility of IT service delivery and emphasizes people and culture and seeks to improve collaboration between development and operations teams while seeking to remove the unnecessary impediments to service and application delivery by making use of agile and lean concepts”

Lean and Agile were extended beyond development to operations to create DevOps.
Why This is Important to You

First and foremost, this is important because it demonstrates that DevOps doesn’t exist in a vacuum. It complements Agile development which continues to focus on improving software development methodologies to promote constant improvement, and Lean management which extend to entire enterprises to encourage environments committed to constant improvement. DevOps applies the same iterative, constant development, constant delivery, and constant integration concepts as Lean and Agile to achieve constant improvement not only in code and the operation of that code, but also of how rapidly customers can get to market and drastically reduce the time in which customers achieve value.

Agile is far more a set of practices, processes, and values that facilitate the execution of software development projects, and other projects. DevOps goes well beyond this and can be considered a philosophy or envisioning of how various functions from disparate departments within an organization could be brought together, augmented with automation, streamlining the progress from design to development to testing to deployment and then back to iterate again. Agile, then, is far more tactical than DevOps.

Other differences between Agile and DevOps include:

» Teams are kept small, when applying Agile, to enable greater speed. DevOps teams include members from various departments across the enterprise, resulting in larger teams.

» DevOps makes maximum use of automation, while Agile just doesn’t emphasize it.

» The anticipated iterative improvement interval in Agile is measured in weeks and months, while DevOps aspires to deliver new code hourly.

» Agile prefers working software over documentation. DevOps leverages comprehensive documentation to help bond development and operations more closely.

A 2013 Gartner report:
» Agile is focused on the development of the software. DevOps concerns itself with fast delivery of the end-to-end business solution. The two are not mutually exclusive.

» Agile uses iterative processes that leverage collaboration, feedback, and small, quick new releases. DevOps goes well beyond that to include delivery, deployment, monitoring, and feedback that lead to the next iteration.

The Future of DevOps

The friction between the Information Systems Development group that creates and improves the software that runs the business and the Information Technologies Operations group that runs the computing infrastructure that delivers those applications has been present for years. The two teams finally saw the advantages in creating a more effective collaboration and created DevOps, compounding their names to signify that they had become one.

Along the way, they left a few other teams out.

Already we’ve seen common usage of the compound word DevSecOps to signify the importance of paying close attention to data and network security at every level, and to incorporate the team responsible for that security in with development and operations. Who knows where that can go next?

Perhaps the realization that the fiscal responsibility involved in creating, operating, and securing the applications that run the business is significant! Might we then extend to DevSecFinOps? While this naming obviously becomes ridiculous quickly, the alignment and collaboration between the various departments and lines-of-business is beyond valuable. Doubtlessly, many successful DevOps cultures will carefully examine what made the initial unity successful and will work to replicate and extend those advantages to every other team in the enterprise.
One Takeaway

The one word that captures the raison d’etre, the reason for the existence of Lean, Agile, and DevOps, is “value.” Whether its enabled by faster and more frequent improvements to software, measured by customer satisfaction levels, or defined by time to market, it is all focused on delivering superior value to customers.

Building a DevOps Culture

More than once has actor Tom Cruise felt “the need for speed.”

In films like “Top Gun” in which the phrase appeared, and also “Days of Thunder” in which he played a race car driver, the most important priority was to get a vehicle to move really fast, preferably faster than any other of its kind.

There’s a stark parallel between this perceived need and perhaps the strongest perceived need in business today – that same need for speed. Every business seeks to stay ahead of its competition by making everything happen faster and hopefully better. Products come to market faster. Messaging about those products come even faster. Everything everybody does must be done faster, so everything that impacts every process must move faster.

Since most things are now run by software, software must move faster and get many things done more quickly. For that to happen, the software itself must be developed faster, deployed faster, in production faster, and continuously improved faster than ever before.
Creating a culture conducive to DevOps can be described with great similarity. Based on the name, it’s easy to assume that DevOps occurs within the team of developers who create the software and the operations group who run the software for users. But the change required is cultural, and cultural change is pervasive across any organization. Everyone is impacted by it. Everyone experiences changes in paradigm, attitude, and approach.

Culture is so ingrained that it cannot simply be changed by caveat. Instead, it is influenced, it is encouraged, and given credible evidence. Then it slowly shifts. As it does, it is important to assure all members of the team that it is safe to accept change, just as it is safe to fail as long as learning results from it.

Acceleration of processes is again accomplished by innovating and improving every component of the value chain. Large monolithic blocks of code that take months to develop, code, and implement give way to microservices in containers that are loosely coupled and much finer grained. These reductions make them far faster to develop, code, deliver, and deploy. Even users are encouraged to perform faster, specifically when it comes to providing feedback for further improvement.

Most of the time, a process that will benefit from automation is quickly automated to achieve greater speed in performing repetitive functions. One of the acceptance criteria for every tool incorporated into a DevOps environment is speed of performance. Faster technology is incorporated at every level from servers and storage to code and compilers.
DevOps Requires a Culture of Trust

Approvals represent a necessary latency in many processes. Approvals take time. Required decision-makers may not be immediately available. In a DevOps culture, decision-making is delegated down as far as possible so that as many people as possible can make a required decision and take productive action to achieve faster progress. Checklists, inspections, audits will all introduce far more latency and destroy the value of any DevOps initiative.

Teams become far more autonomous, more enabled, and more empowered, able to make decisions and create needed changes without waiting for someone else to go through their seemingly endless deliberations. At the same time, they also become more interdependent as they share responsibility for processes that cross from one to another. The crucial role of feedback in driving the next iteration of improvement only serves to further illuminate the importance of the high level of communication and collaboration that DevOps organizations must foster.

The growth-oriented mindset present in a DevOps culture tends to toss out the old rules. It’s okay to fail, as long as you fail quickly, learn, and recover quickly. The ultimate goal is getting quality working software delivered quickly and often. Any rule that delays that must be ignored or changed.

Another manifestation of trust is the elimination of blame. It really is unimportant who may have caused a particular failure or anomaly. Finger-pointing, is usually replaced with a united desire to rectify anything that may have gone wrong quickly, and without rancor.

The only important thing is to rectify the situation and provide quality working software as quickly and as often as possible.
DevOps Requires a Focus on Outcomes Rather than Control

It was in the aforementioned “Days of Thunder” that Nicole Kidman turned on Tom Cruise and declared that “control is an illusion, you infantile egomaniac!”

In a DevOps culture, “control” must be distributed. Teams become self-driven and self-managing. Improvement is controlled through testing with the desire to improve a pervasive element of everyone’s approach to their mission. Perfection is happily sacrificed on the altar of good working software that will quickly be further improved. The outcomes are what matters, not adherence to process or compliance with requirements. Constant improvement is exercised through user feedback.

Indeed, in an interdependent, cross-functional culture like DevOps, there’s seldom certainty as to who is in control. It becomes attractive to cede as much control as possible to automated systems to eliminate any possible conflict. But automated tools, and automation in general, are not the sum total of DevOps.

Automate That Which Can Be Automated. Share Everything.

“Even with the best tools, DevOps is just another buzzword if you don’t have the right culture,” writes Martin Fowler, self-proclaimed author, speaker, and loud-mouth on the design of enterprise software. “An attitude of shared responsibility is an aspect of DevOps culture that encourages closer collaboration. It’s easy for a development team to become disinterested in the operation and maintenance of a system if it is handed over to another team to look after. If a development team shares the responsibility of looking after a system over the course of its lifetime, they are able to share the operations staff’s pain and so identify ways to simplify deployment and maintenance by automating deployments and improving logging.”

Continuous Improvement is the Product of Culture

Agile methodology is executed in short “sprints”. By comparison, DevOps is a marathon that contains many sprints. It requires stamina and determination to remain fixed on the ultimate goal of constant and continuous improvement.
“True DevOps unites teams to support continuous integration and continuous delivery (CI/CD) pipelines through optimized processes and automation,” says Waleed Bekheet, VP of Practice Solutions at global technology consulting company Onica. “A CI/CD approach enables efficiency in the building and deployment of applications, and automated application deployment allows for rapid release with minimal downtime.”

It may seem like a panacea, and the cynics among you may call it unrealistic, pie-in-the-sky, fluffy, or some other term that is derisive of new age thinking, but achieving continuous improvement is not that. It is the product of a culture created by the desire to do everything necessary to move from idea to implementation to increase success faster than ever before.

It seems obvious to observe that the benefits of implementing a DevOps environment will inure to the development and operations departments and that makes great functional sense.

Beyond development and operations, DevOps’ benefits must also increase value for the business. In the heat of daily activity this can sometimes be forgotten or overlooked. For the overall DevOps initiative to be successful it must be at the center of all planning and execution. DevOps’ value must be found in the context of the business.

**The Benefits of DevOps**

**Agility**

Agility, in the context of development, is created by rapidly receiving frequent feedback from users and having the ability to quickly respond to the feedback with improved code. Their results aren’t actualized until operations deploys them, so their agility is enabled by obtaining and collecting feedback which they can immediately share with development and very quickly see the response to that feedback in the form of new and improved code. The speed at which both achieve agility is directly dependent upon how efficiently they can each perform their tasks. By using the same tools and platforms many of the impediments between them are removed, making for much more frequent iterations of constant improvement.
All of this underlies the agility the business needs to achieve. Market conditions, competition, supplier changes, regulatory action, and many other conditions are impacting the business at an incredible rate. The ability to evaluate and respond to all these changes quickly is critical to remaining competitive and, ultimately to remaining viable. For the business, this is the definition of agility enabled by DevOps.

Alignment

Implicit to the efficiency of development, operations, and business working together is the important process of keeping them in alignment with each other. This isn’t just strategic alignment in which everybody agrees on how to move. This is tactical, day-to-day, pragmatic alignment that absolutely must be constant, consistent, a shared commitment. Delays in developing code cost the process. Operations delayed in delivering and deploying new updates cost the process. Users taking too much time to provide feedback cost the process.

Taken together, this demonstrates the importance and value of DevOps in how it promotes interoperability. Using consistent policies, consistent procedures, and a shared responsibility to accelerate the process in every way possible brings teams together far more effectively than any team-building exercise possibly could.

Risk Mitigation

Risk is very familiar to all IT operations, which lives with the constant risk of business disruption and delay caused by undetected development errors. This creates substantial financial risk for the business and the risk of lost time and productivity for staff. By breaking the software development process down into far smaller pieces all these risks are mitigated. When rollback or resolution can be almost instantaneous, all these risks are reduced or eliminated altogether.
Innovation

Anything that doesn’t continue to grow and improve eventually dies. The constant improvement that is the major goal of DevOps is fueled by innovation. Feedback from users to development through the collection activities of operations does more than just resolve problems. The talented people who are closest to the activities that drive the business are most likely to generate new ideas, new concepts, and new value to be incorporated into the next iteration of the code. This means that everyone involved in the DevOps process contributes constantly to the process of innovating.

Velocity

Given that software drives every business, it follows that:

» Faster time-to-collecting-and-implementing feedback leads to...

» Faster time-to-improved-code which leads to faster time-to-production which leads to...

» Faster time-to-market and, ultimately, much...

» Faster time-to-value for the business and its customers which means that the business is exhibiting...

» Faster time-to-respond to all the constantly changing market and customer conditions

Improved code does not only refer to faster processing, it also includes constant improvements to the quality of the overall software and the stability of all operations to avoid costly anomalies or outages.

The time value of money has never been more evident than it is today. Whoever employs the best software first earns a tremendous competitive advantage. DevOps’ reason for being is to develop better software faster.
Process Optimization and Constant Re-engineering

Bringing development, operations, and users together to align their working toolset enforces constant observation and evaluation of all processes. This evaluation leads to constant improvement through innovation and response to feedback. As such it instills built-in constant process improvements by constantly re-engineering all processes with full participation and agreement among all participants.

Productivity and Cost Reduction

The benefits most often cited for various process improvements have been saved for last in this discussion. The feedback that users provide comes from their need to improve their productivity. Since this feedback is the fuel that runs the DevOps process it is the pursuit of improved productivity that drives everything in an operation. Finally, cost is reduced by saving time at every stage, which contributes to the increased return on investment that is the ultimate objective of every business.

Businesses in today’s climate of digital transformation depend upon software as one of the foundational pillars of their strategy. Anything that contributes to making and deploying better software faster results not only in the business’ ability to survive and thrive. It results in improved strategic competitive advantage, greater investor and stakeholder value, greater ROI, greater business valuation, driven by highly enabled user communities.
Understanding the Challenges of DevOps

It is as important to consider the possible disadvantages of implementing DevOps in your organization as it is the benefits. The decision to implement DevOps is by no means trivial.

This is the first challenge of DevOps today, driven by its relative newness. Many definitions of DevOps have emerged, and it is important that an organization achieve agreement as to what their definition will be. This agreement is informed by several things that DevOps is often thought to be but is not.

It’s not just a methodology. DevOps goes well beyond the requirements of a new methodology to include organizational, cultural, procedural, and other positive changes. In fact, DevOps holds the potential to provide such benefits beyond just the development and operational teams, but throughout the entire organization? Thought must be given to how to get the organization ready for such broad, beneficial change. Is the reasoning driving the decision of great value to you? Who will benefit most? Who will require support to achieve the necessary changes?

Technology investments in the automation tools required for DevOps can be costly and take a great deal of time to identify and more to implement. There is also a tendency to implement various “best-of-breed” tools with little or no consideration for how well they interoperate. The pain of changing tools midstream has taught many the value of establish a “toolchain” that does incorporate high integration and interoperability.
Before the beginning of your DevOps implementation project you’ll want to map out standards within your organization and optimize workflows. In addition to there not being one globally accepted definition of DevOps, there is a standard set of practices, though processes and procedures will vary based on business needs. Many roles may be refined or redefined. Processes and procedures will be adapted, new ones developed and deployed, and training provided as required.

**DevOps Requires Culture Change**

It’s commonly observed that you cannot just change a company’s culture on command. You can influence the culture, shift it, it can evolve over time, but it’s nearly impossible to just instruct all employees to simply change the way they think and act about specific things. The culture of any organization starts at the top of the leadership hierarchy and trickles down throughout, filling every empty space.

It’s easy to think that the only people impacted by the implementation of DevOps are the developers who write code and the operators who run that code on the network for the users. But those teams do not exist in a vacuum. The code they write and run serves other members of the group in departments throughout the enterprise. The process is dependent upon those users for the feedback that enables rapid iteration and constant improvement. Since this is a crucial element to the success of a DevOps process they must be ready to absorb the new culture as well as anybody else.

Another benefit of DevOps is that it may break down and unify silos of information and applications. There are also silos of people stuck in their own processes and ways of managing information that should be brought closer together through the implementation of DevOps.
For any culture-impacting change to successfully occur there must be “buy-in” at the top of the organization. For DevOps initiatives to be successful they require “buy-in” from everyone. This kind of all-encompassing whole-organization culture-change can be very disruptive to even the most well-developed companies. That easily becomes an overwhelming disadvantage.

**DevOps Does Not Solve Every Problem**

If your organization only requires software upgrades occasionally, and the speed at which they arrive is not mission-critical, you may have no need to stress your operations to implement DevOps.

There may be less expensive ways to achieve the same results. Good DevOps experts are in big demand and are, therefore very expensive. That’s if you can find them. The automation and other software tools they require, and other operational costs add up quickly to a very expensive initiative.

The challenges and potential disadvantages facing those considering the implementation of DevOps generally occur within four categories:
Organizational

Traditional developers worked hard to become proficient with the “full-stack” of protocols and procedures required to develop software. DevOps breaks those multi-disciplinary tasks down assigning each one to dedicated specialists. Over time it will become difficult for those full-stack experts to leverage their many skill sets. At the same time, the organization may find that it has hired and is managing a greater number of people, each assigned to one of a list of tasks that were previously all handled by one.

Also, while the concept of constant improvement through rapid iteration of upgrades sounds fantastic, many users find it difficult to keep up with and start taking advantage of new features as they emerge. Focusing more on delivering working software than on documentation is preferred by both DevOps and its foundational Agile roots. Maintenance of a well-groomed backlog will help developers and operators to keep track of their advances.

The entire organization is involved in any DevOps initiative, and the adoption curve may be filled with dramatic ups and downs. One important way to mitigate this potential downside is to identify a system to track everything, including every task and its outcome, and to facilitate rapid feedback between users, operators, and developers as early in the project as possible.

Processes

One of the characteristics of processes introduced in a DevOps initiative is the sheer speed with which feedback is requested and responding changes are made. This often eliminates all opportunity to “carefully consider” suggested changes as everyone could do before. It will take time for users to adjust to this change in velocity.

Procedurally, the organization will need to be more fault-tolerant at least in the early stages. Mistakes must be made and observed to stimulate rapid learning and resolution. This will be uncomfortable to some.
Testing cannot become a casualty of DevOps processes. Instead, faster methodologies must be adopted to fulfill the objectives of rapid iteration and constant improvement. This usually turns into automation of many testing processes which should result in different teams being able to make needed changes rapidly.

**Technology**

There’s a tendency for people to say DevOps and think automation. DevOps is not just automation, though automation enables evaluation and analysis that leads to improvement quickly enough to satisfy the velocity needs of every DevOps process. Indeed, the most important segments of the DevOps workflow to automate involve the feedback loop back from users to developers and quality assurance testing.

**Speed and Security**

Speed and security don’t necessarily get along with each other very well. It is critical to the success of your DevOps initiative that you carefully consider a separate plan for security at every stage of your DevOps workflow. Any chain is only as strong as its weakest link, so it is critical to assure that the security of your DevOps processes is consistently strong across the workflow chain.

Security teams traditionally begin by evaluating the design of a system to determine weak points, choke points, bottlenecks, and other areas of vulnerability. Remember that DevOps inherited from Agile the goal of achieving a minimum viable product (MVP) as quickly as possible, and then preferring iterative improvements of working software over documentation. As such, there really is often no formal design and therefore no design documents for anyone to evaluate and analyze. The security team must then depend upon the advice of the developers and the operations people. This promotes much better teamwork but challenges the achievement of sufficient security from the start.

Because of this, and because neither developers nor operators are necessarily security experts, we have seen DevOps quickly evolving to become DevSecOps.
Turn the Challenges of DevOps Adoption into Insights

One of the main goals of DevOps is to be first to market with a new idea or feature. This requires maximizing efficiency and eliminating any waste, process delays, or unnecessary costs. The desire is to fail-fast-fail-early and resolve everything as quickly as possible.

Our final daunting challenge is the breaking down of silos. Certainly, the amalgamation of silos of data, but also silos of personnel in various teams, each with their own standards and process preferences.

Implementing DevOps: Adopting and Adapting to a Culture of Collaboration

Since a major goal of DevOps is to create an environment of continuous improvement through continuous development, the implementation of any DevOps initiative must be seen as an ongoing, never-ending process that impacts far more than just any one given project. It becomes a unifying way in which the various departments within a company learn to collaborate more effectively.

This requires an organization-wide mind shift, not just a paradigm shift, but a holistic change in the way everyone thinks about how best to make improvement happen. Many have referred to their initiative as a DevOps “movement” to signify the necessary winning of hearts and minds.

The greatest challenge for those approaching the implementation of DevOps in their organization will be the need for patience.
The concept of constant iterative change is not readily or easily embraced. It involves constant evaluation, which can be anathema to many members of the team. Actions are viewed and taken with far greater granularity.

The initiative must start small and prove itself, then be distributed organically with full buy-in from each department that becomes involved. It may take two or more years before substantial benefit becomes obvious. Then there’s the other challenge:

DevOps is still in the process of being defined, and everyone has their own interpretation. While many “best practices” can be suggested, it is up to each reader to determine what is “best” in their particular environment. As such, the following are suggestions for your consideration.

“DevOps is not a goal, but a never-ending process of continual improvement”
– Jez Humble, one of the progenitors of DevOps

Stay CALMS and Begin with the End in Mind

One of the frameworks used to evaluate an existing DevOps implementation is CALMS, an acronym standing for Collaboration, Automation, Lean, Measurement, and Sharing. Beginning your DevOps planning with this metric in mind helps keep the focus on implementing DevOps in its entirety, from:

» Developing a pervasive culture of collaboration.
» Implementing effective tools and automation.
» Doing everything to keep everything as quick, small, and minimal as possible.
» Assuring that everything is measurable so it remains manageable.
» Breaking down silos wherever they are found and sharing data, processes, tools, everything.
Remember that Culture Change is Hard

For your DevOps implementation to be successful, it must create significant change in your company’s culture. This is no simple task. Culture is not conveyed in rules, regulations, processes or procedures. You cannot document culture. Culture instead pervades an organization from its leadership on out. This means that your DevOps initiative must begin with full buy-in from the top. Change to the culture is significant, organic, gradual change.

Most people resist change, especially when it comes to the culture they are accustomed to living in. While implementing DevOps involves humanistic change for the better, it’s still change. Always remain sensitive to the natural resistance you will encounter and find ways to overcome it gently, gradually, and productively. While you’re at it, work at removing any “blame game” tendencies in your existing culture as you make your changes.

You may find it valuable to create a core DevOps team with identified ambassadors or champions in each department involved.

Implicit in the process of changing the culture is to start by changing the way and the frequency with which people communicate. You’ll need to rethink how much detail is transferred from one department to another to facilitate more effective collaboration. The foundation of improved communication is the standardization of how everyone communicates on the same platform, preferably a robust platform that supports text and email messaging, voice, video, screen sharing, document collaboration, and more.

Always keep in mind that most everything you’re doing challenges how people have always done what they do, and that’s going to be uncomfortable at first. Where they formerly were responsible for the one specific thing they contributed to a process, they are now being asked to share ownership of the entire project. Even those who initially adopt will backslide. Persistence cannot be allowed to depend upon each individual. Whoever is driving the DevOps initiative must make provisions for supporting and helping people through their own individual transition challenges.
Focus on People and Their Needs

DevOps functions in the context of your company and its business. Many companies adopting DevOps do so because they need to bring products to market more quickly. They are willing to embrace the constant iterative nature of DevOps processes to get to market faster, but that is not the only need involved.

Every step along your roadmap must be justified by what your business needs, and you must obtain agreement by the stakeholders that they agree with the needs you’ve identified, and that they’re prepared to adopt a new culture and adapt to the changes it brings. Again, this usually proves to be far more difficult than anticipated.

You’re not implementing DevOps because its “fashionable” so remember to define business needs in the context of your customer. High customer satisfaction always remains a key goal of any process and DevOps is no exception. Without customers, it hardly matters what you do or don’t do.

Remember that your own people are being asked to significantly change the way they do what they do, which may have remained the same for a long time. DevOps implementation is not forced upon people, or the optimum velocity of iterative improvements may never be achieved. Team satisfaction is at least as important as customer satisfaction in successfully integrating DevOps into your environment.

Remove All Barriers

Kishore Jalleda, Sr. Director of Production Engineering at Yahoo says “DevOps is really about eliminating (most) Technical, Process, and Cultural barriers between Idea and Execution — using Software!”. 

To create and maintain the most fertile environment for cooperation, great emphasis should be placed on removing as many barriers between people as possible, whether they be physical, emotional, moral, ethical, interpersonal, or anything else that impedes collaboration and cooperation.
Even training and orientation should include people from other disciplines to promote better understanding and appreciation of each other’s realities. Have operators attend development training and developers learn more about operations. This can only promote greater consanguinity. Rather than characterizing this alignment as getting everyone to march to the beat of the same drummer, its more nuanced than that because it pervades literally everything everyone does and how they do it. It encompasses intention as well as action. Truly it is more like getting everyone to dance to the same tune in the same rhythm. When its working, everyone and everything resonates.

Tools and Automation

The easiest way to make things, especially repetitive things, happen faster is to automate them and the tools that help you do so are many. Automate everything that you find can be automated and allow people to focus on more important issues. This will accelerate everything. Encourage everyone to share the tools they’ve found with the DevOps team for possible deployment.

This last point cannot be overstated. The goal is not to first identify all the tools and automation opportunities. The goal is to institute process change focused on increasing customer and business value. The process of seeking and identifying tools will be greatly enhanced when the people involved are already living with DevOps principles.

Paving the Path for DevOps Adoption

The failure of users to adopt any new platform is often cited as the underlying cause behind three-quarters of new initiatives. Given that DevOps requires deeper commitment by a broader variety of people than most platforms or methodologies, nowhere is it more critical to pay close attention to the active promotion of adoption than it is when transitioning to DevOps.
Users and even executives cite several reasons why they fail to adopt a DevOps culture-shift. Many suggest that other technology and business initiatives are higher priority to them than DevOps. Some point to a shortage of resources impacting the decision to change. Some simply don’t see value in making such a comprehensive change.

**Changing the Culture**

Again, culture is by far the hardest thing to change in any organization. You cannot request it, assign it, or demand it. It must permeate the organization top down. This is a process that doesn’t involve talking, imploring, demanding, or begging. None of those will ever change culture.

Giving evidence, demonstrating value to be obtained, celebrating successes, these are the elements that lead to culture change.

This challenge is exacerbated by the sheer scope of change you’ll need. Just about everyone in the organization will need to be reached and persuaded to alter their paradigm and accept a new culture of continuous improvement through continuous development, rapid feedback cycles, constant upgrades, and more.

DevOps generally requires a constantly changing culture. The cycle that leads to continuous improvement is iterative. Each cycle improves upon the last and sets the stage for the next. Analytics, rather than human proposals, will drive many investments. Changes will be incremental but constant iteration will outpace prior version changes.

Influencing and shifting the culture begins with the alignment of the way in which everyone thinks about how they do what they do, and how it contributes to the greater whole as people give up the traditional ways in which they have done things. Commitments need to be made that they’ll be supported in their efforts to learn and adopt new ways. As this begins it may seem deceptively easy, but don’t be deceived. The “honeymoon” will soon end and much more effort must be put in to sustain the transition.
Overcoming Inertia

In this case we’re referring to the tendency of a body at rest to remain at rest. People resist change. Some even fear it.

DevOps benefits from the incorporation of Artificial Intelligence (AI). Many team members fear this. Some will have the attitude that legacy applications still work, resulting in "it ain’t broke, so don’t fix it" thinking. Implementing DevOps requires new tools, techniques, and infrastructure which disrupt existing investments and require retraining.

Perhaps the most vivid example of resistance to change will come from the adoption of applications built using microservices in containers which is required for the success of DevOps as it delivers a superior ability to iterate improvements. People accustomed to monolithic applications may not be immediately impressed by the transition, preferring to focus on the uncertainty.

Do not underestimate the passion with which some will work to maintain the status quo. These doubters will require profound evidence of potential positive gain for themselves. Many of these will be at management levels that can blockade their entire team, keeping them in their own silo.

One great truth of human nature to depend upon is that people will usually be far more accepting and enthusiastic about change the more clearly they can see how it will benefit themselves.

Traditional Disconnects

The core driver of the development of DevOps was to overcome the traditional friction between developers and operators of software. Traditionally, neither has considered the other’s requirements when doing their job. The result is a focus on blame, finding someone to blame for errors or omissions. Investing more in blaming than resolving always results in endless rework, frustration, and argument. A truly functional DevOps environment is one in which it is safe to fail, because failures promote constant learning and improvement.
Developers tend to assume that the resources their software needs will be available. Operators are often taken by surprise and must scramble to provide many resources. Each may end up resenting the other for making their life more difficult. The sense of ownership of the software is never shared. DevOps is intended to overcome and reverse these, putting all participants on the same page.

**Priorities**

The fact that you are properly proceeding gradually as you roll out DevOps will almost inadvertently upset some people. They want to see immediate improvement, but now have to wait until the DevOps deployment team gets around to them. The fact that applications must be evaluated before they are improved will frustrate many, driving them to attempt to alter the DevOps leaders’ priorities.

Anticipate friction. Remember that everything about a DevOps strategy is ultimately meant to be customer-focused, not product-focused.

**Infrastructure Changes**

The traditional response to most IT problems has long been to “throw more infrastructure at it.” But that only makes IT operations more expensive and less effective. DevOps leverages procedural and process changes to create new efficiencies, new economies of scale, greater performance, requiring modernization and optimization of multiple systems to enable constant development and constant improvement.

**Ownership**

The importance of the sense of ownership of the change and its impact on adoption cannot be overstated. Not only ownership of the change itself, but a sense of ownership of the planning for the change, and participation in the decision to adopt DevOps. When the teams participate and feel a sense of ownership of DevOps implementation they almost automatically adopt.
The DevOps Lifecycle

The concept of joining development and operations people together support the goals of DevOps:

- Improve collaboration between all stakeholders from planning through delivery and automation of the delivery process in order to:
  - Improve deployment frequency.
  - Achieve faster time to market.
  - Lower failure rate of new releases.
  - Shorten lead time between fixes.
  - Improve mean time to recovery.

This is accomplished by implementing and adhering to a straightforward lifecycle of activities:

- Continuous Development.
- Continuous Integration.
- Continuous Testing.
- Continuous Monitoring.
- Virtualization and Containerization.
Clearly DevOps fosters a culture of continuous everything! The result is an iterative cycling creating constant improvement with far shorter update cycles that accelerate time to market while making it far easier to identify and correct deficiencies should they occur. That iterative cycle repeats constantly:

- **Be Agile**
- **Plan**
- **Build**
- **Test**
- **Release**
- **Configure**
- **Monitor**

From the monitoring, we learn what we need to know to return us to being agile and beginning the next plan. Continual improvement.

**A Lifecycle of Continual Improvement**

Most of us are accustomed to periodic updates and upgrades. New features, new capacities, new security provisions are introduced that make our software experience better. Some of these are ad hoc, distributed when a need, such as a security weakness, is identified and must be addressed immediately.
Most updates are distributed weekly or monthly. Upgrades perhaps a bit less frequently. Major new version introductions may take years.

These are true for software applications that have been developed in the classic, monolithic way. Any change must be planned, designed, coded, prototyped, piloted, tested and re-tested before distribution.

In cloud environments, monolithic application development no longer makes sense and has been superseded by microservices delivered in containers along with all libraries and other resources required to run them. These microservices are called upon as required. Should one become damaged in the course of transit or processing, it is instantly re-instantiated and re-distributed. This delivers a level of resilience unavailable in monolithic development.

Similarly, today’s DevOps Lifecycle re-emphasizes the concept of “cycles.” Here, the cycles come far faster and more frequently. In DevOps, the developers and operators that have joined forces develop a culture in which they are constantly finding new, better, faster ways to work. Leveraging their foundation in Agile development methodologies, they build new code in collaborative environments rather than silos, and they focus on “short yardage,” creating repetitive rounds of incremental improvement.

As soon as the developers have distributed new code the operators step in and start obtaining user feedback. This vital information is fed back immediately to the developers who then assess the input, create code to build new solutions. Since each is a small component of a much larger environment they can be tested, packaged into their containers, released, configured, and distributed at which point the operators step in again and start obtaining the next cycle of user feedback, starting the next cycle. This repetition continues, and continues, in a lifecycle of constant, continual improvement.
Iterative Lifecycle Process

Even though they are now working together collaboratively, it is useful to examine the steps both “Dev” and “Ops” execute in the course of the DevOps Lifecycle. Let’s enter right in the middle.

Ops Task Cycle

Monitor-Audit-Diagnose-Tune-Feedback

A major responsibility that falls to Ops is the constant monitoring of applications to assure optimum performance. This cannot end at simply monitoring. Going deeper, Ops must regularly audit the system surrounding each application to assure that everyone is doing what they are supposed to be doing in a timely and efficient manner. As they expose weaknesses, they must diagnose the underlying root causes. If possible, they will tune any configuration parameters that are available to them in order to resolve the anomaly. If they are not able to do so, the report of this anomaly becomes part of their regular feedback to Dev.

Dev Task Cycle

Analyze-Edit-Build-Test-Debug-Deploy

As new feedback comes in, Dev analyzes it to determine root causes and identify opportunities for improvement. They may then edit existing code or build new microservices to resolve reported issues. Once corrected or conceived, the changes and additions must be tested and then debugged as necessary. Finally they deploy the new microservices at which point Ops begins monitoring them.

And the next iteration of the lifecycle continues. Again, and again, and again. Continuous development. Continuous Integration. Continuous Improvement.
DevOps Tools

Think back to the introduction of any major new technology innovation and you’ll recall an explosion of tools and utilities surrounding that product or process. Eventually the strong survive, consuming some in the middle, and everyone else moves on.

DevOps is no exception. To span both development and operations several tools will obviously be required. The challenge as always is to choose the right ones to invest in the first time so you can preserve your investments as you grow. It is as difficult a challenge with DevOps as it has ever been with any emerging methodology.

Tool Categories

An excellent way to begin to approach the selection challenge is to break the big problem down into smaller component pieces. In this case breaking the many tools now on the market into categories to create smaller groups of choices to consider.

Many of these categories correspond closely to the stages of the lifecycle discussed earlier. Others are environmental tools meant to support a more conducive way to weave development and operations together:

Collaboration

The most important connections are those that are made between members of the team. Communication between team members must be as rapid and effortless as possible. Many DevOps collaboration tools make it as simple as a chat to start. When the situation warrants, the chat can be easily replaced with voice, and video, and screen sharing, calendaring, white boarding, and many other convenient features. Perhaps the most valuable is the ability to have several team members simultaneously edit documents.
Code review

Just as no writer can properly review and proofread their own prose, developers are ill-served to review their own code. People tend to see what they expect to see which makes it very likely that a developer will pass over the obvious logic errors they wrote. Review by a colleague avoids this problem. They will also determine whether all of the functional specifications and requirements have been fulfilled and that the code conforms to the guidelines established for the project. Accelerating the process will be automation tools to perform some of the standard tests. Code review tools also provide important version control.

Continuous integration and continuous deployment (CI/CD)

Welcome to the core purpose of DevOps, continuous improvement through continuous development, continuous integration, and continuous deployment leading to continuous feedback that starts the entire cycle again. By definition this is an iterative process that keeps adding value with each cycle. The flip side is that every cycle also adds complexity. CI/CD tools exist in many flavors that each help manage the growing complexity. Some automate software testing and deployment. Others focus on integration and delivery. Some feature plug-ins that will automate just about any task in a DevOps environment, or at least claim to.

Build automation

Development is a team effort with many contributors. Once a developer has created source code, it must be retrieved from the code repository and compiled into machine language by a build script. The result will be integrated into the shared environment where it will interact with code produced by other developers. Before that happens, it is best practice to ensure that the new binary code will not impact existing code negatively. Build automation tools perform this testing. It also supports linking of modules and processes, documentation, testing, compilation, packaging, compression, and distribution of binaries.
Testing automation

DevOps accelerates the process of developing software, testing it, deploying it, and returning feedback quickly to move to the next iteration. The concomitant commitment must be to deliver “quality at speed.” Faster isn’t better if the quality is absent. Testing is often seen as a prime source of latency in the process, so test automation tools have been developed to automate as much of the code testing process as possible to reduce that latency.

Release automation

Also referred to as Application Release Automation (ARA), release automation describes the process of packaging and deploying an application or updates to an application. It spans the entire workflow required to get the software to production. This very effectively supports the DevOps goals of more frequent releases at greater quality.

Configuration management

Consistency is a key characteristic of a quality DevOps environment. Servers, storage, networking, code, operating systems, literally everything every step along the path must be kept as consistent as possible to optimize the speed of all workflows. At the same time, consistency enables scalability. When every configuration is identical it becomes simple to replicate them rapidly. Any manual intervention immediately introduces latency. Configuration management tools are used to facilitate version control and to make changes and deployments faster, predictable, scalable, replicable, keeping all controlled assets in their desired end state.
Infrastructure and application monitoring and management

By uniting development and operations, we acknowledge that it’s not enough to iteratively develop and deploy software faster. We must also assure that the software runs efficiently and constantly. Application Performance Monitoring/Management (APM) constantly monitors all of the resources required for applications to run optimally, alerting operators or, preferably, taking automated resolution action themselves. Similarly, the underlying network on which everything runs must be similarly monitored and managed. Reports generated from both systems enable system owners to identify proactive steps to take that will improve operations.

Containerization

Containers make DevOps development and delivery easier, because they assure that the operating environment the code is running in doesn’t change between developers, between environments, or ever. Containers accomplish this by packaging everything the application needs, the code and all its dependencies, including runtime, system tools, system libraries and settings, inside the container, conveying ownership of the entire package that actually runs the software upon the developers. This makes it far easier for the developers to share the complete package with IT operations, the defining goal of DevOps.

Serverless Computing

Prior to the introduction of “serverless” architecture, developers needed operations to provision and maintain servers to provide the runtime services they needed to build and test code. Serverless simply moves that to the cloud. Literally every task performed in a DevOps environment is accelerated by the constant availability of consistently configured resources from development to testing to deployment. This dramatically improves the role of Operations as it removes what is probably their most time-consuming, commoditized set of tasks. Thus, they are freed to engage in far more valuable activities to support development and the user community.
DevOps Toolchain: Innovate Faster and Deliver Superior Software

We’ve introduced and described categories of tools rather than highlight specific tools, partly because there are so many of each category that we could never cover them all and this list becomes a moving target as the DevOps movement evolves.

This should raise the question in your mind, “how are we going to select from them all?” and there are plenty of analysts attempting to answer that question. If each of these categories consisted of only ten choices, that would involve evaluating 100 products. There are far more choices in the market and new ones entering every day.

This is a great driving reason to select and establish a complete end-to-end toolchain of selections including each of these categories. Ideally, using each tool for each stage in the chain culminates in starting to use the next tool in the chain. DevOps is obviously not this linear, but progress forward is always a good model to base strategy upon. Of the many advantages to be gained from having a unified toolchain, the simplest to understand and most fundamental is serving the competitive need to innovate faster, delivering superior software faster and more frequently.

Improving exception handling and incident management is also key to maintaining high velocity in software delivery cycles. Ultimately, this will also serve to help you identify and resolve defects.

A DevOps practice doesn’t necessarily need tools, and is certainly not defined by them, but given that the goal is accelerated release of constantly improving software, anything that speeds up the processes inherent in DevOps is, by definition, a good thing.
Avoiding Latency

In DevOps, latency is our enemy. The idea is to accelerate the cycles, so anything that slows them down hinders us.

Using tools that don’t work well together introduces latency, usually in the form of workarounds devised by developers and operators who all have better things to do with their time than devise workarounds.

This adds a whole new level of complexity to our tool selection problem. Not only do we need to evaluate each tool, comparing it to the other choices in their category, we also need to gauge the interoperability between them and tools in other categories.

Consistent with DevOps philosophy we also need to determine how acceptable each selection will be to all members of the development and operations teams, since the tools’ interoperability will help determine the teams’ interoperability and seamlessness.

So our challenge is to identify best-of-breed tools in each category, proving that they all work well with each other, and will all be fully acceptable to all involved parties. This will not be simple.

Alternative Strategies

One strategy that most every organization explores for just about every initiative is to mandate adoption. This almost never works well. Departments are not only invested financially in existing tool platforms, they are also pragmatically invested in training, policy and procedure development, and more. The one thing that results from most mandates is resistance.

At the other end of the spectrum would be a wide open environment where everyone gets to choose and use their own tools. This results in silos of operation, one of the things we most want to eliminate.
A truly extraordinary team could work together to identify tools that will work well for all of them and with the other tools they choose. Recognize that this will likely consume significant time and resources.

Most often, the best strategy will include sufficient planning to promote rapid user adoption of all tools selected.

Integrated Platforms

Most of us are accustomed to platform providers offering up an integrated suite of related services. It is certainly the rule in Enterprise Resource Planning (ERP) software and productivity suites. The disadvantage may be that some of the modules aren’t the best-in-class, so you might have a great word processor in your productivity suite, but the spreadsheet doesn’t beat the competition. The ERP order processing may be extraordinary, but the inventory management leaves much to be desired.

The big benefit of using an integrated platform comes in their high level of interoperability. It’s easy to copy a spreadsheet and paste it into a document in the word processor. Order Entry and Inventory both inform General Ledger flawlessly.

DevOps Automation

It is tempting when implementing DevOps to seek to automate everything you can find that can be automated. This is a temptation to be resisted.

Great automation starts with great manual processes. When you take an ineffective, flawed manual process and automate it, you’re automating a bad process.

The process remains just as bad as ever, it just happens faster and more frequently.
When you’ve determined that your manual process is highly optimized and effective, it’s time for someone to step up and suggest that process be automated.

The Path of DevOps Automation

Consider automation to be a journey that has a definite beginning and an unclear ending. Unclear because great automation of DevOps, by definition, should end in continuous iteration of development cycles that yield constant continuous improvement.

Begin by gradually automating development processes that will yield optimum improvement. Anything that’s repetitive, especially if it’s banal and routine, is a candidate for automation. Once you’ve identified your automatable process, begin searching for the right tool to accomplish it.

This is an important point. Many see tools or vendors of tools who work to convince them that theirs is the best possible tool for the purpose. The problem is that there’s no such thing. Every environment is different, and different tools will suit different processes in each environment differently. It is all too easy to fall into the trap of selecting a tool and then finding processes to automate with it.

It is usually far more efficient to plan automation and select tools based on sequences of tasks. For example, as we focus on development, the sequence might be:

- Develop code
- Testing the code
- Deployment of the infrastructure for testing
- Run tests
- Deploy the code upon passing of all tests
- Collect production metrics
- Orchestrate the DevOps pipeline and return the feedback from production
Benefits of Automation

Just about everything is benefitting from automation these days, and DevOps is no exception. As with most process-driven environments, automation enables speed, scale, consistency, and feedback. All four are critical to the success of an iterative constant-improvement environment like DevOps.

**Speed**

Every manual process that is automated enjoys significant increases in speed, so the more automated the processes the faster the overall engine becomes. Add to this a dramatic reduction or elimination of user errors.

**Scale**

Many manual processes simply cannot scale because humans cannot execute the necessary tasks quickly enough to complete them in the time allotted. As data entities grow larger, the need for automation grows along with them. Automated systems have far greater elasticity, so additional resources can be provisioned as they are required to accommodate scale.
Consistency

Automated processes execute exactly the same way every time they execute. They do not vary. They are not subject to external conditions such as mood, distraction, or other human elements. Consistency of operations brings control, which reduces risk and adds speed.

Feedback

In a well-automated environment, software can request feedback from users while they are using it. Many information-gathering efforts happen with greater assurance and consistency when they are automatically executed. This feedback in turn informs the next round of changes needed which contributes powerfully to the goal of achieving constant improvement.

In a well-automated environment, software can request feedback from users while they are using it. Many information-gathering efforts happen with greater assurance and consistency when they are automatically executed. This feedback in turn informs the next round of changes needed which contributes powerfully to the goal of achieving constant improvement.
Measuring DevOps: Proving ROI on Your Investment

Legendary management teacher and thinker Peter Drucker told us, “You can’t manage what you can’t measure.” He later paraphrased himself to say, “If you can’t measure it, you can’t improve it.”

This advice is not lost on those who are implementing DevOps in their organization. Given how all-encompassing the effort is to do so, it is important not only to management but to everyone involved to be able to point at significant improvements and significantly increased value derived from it.

Many analysts and pundits suggest that DevOps is difficult or even impossible to measure, but others choose to invoke the sage advice in one of Stephen Covey’s 7 Habits of Highly Effective People, “Begin with the end in mind.”

Define Your DevOps Goals and Objectives

We measure to determine our progress toward specific goals and objectives. The goals tend to be somewhat subjective and focused on achieving quality improvements. You’re going to want to assess what you’ve learned from the constant feedback received from users. How many of your users are embracing the change to DevOps? How are we evaluating the accuracy and applicability of the changes we are making in end products? How impactful have our efforts at streamlining processes really been. None of these subjective evaluations are easily measured with great accuracy.
The objectives are, just as they say, much more objective and empirical. Easier to quantify.

**Continuous Improvement**

Continuous Improvement is most often cited as the primary goal of any DevOps initiative. One of the commonly cited of the many definitions of DevOps says, “DevOps is the delivery of application changes at the speed of business. By replacing the infrequent updating of monolithic blocks of code with very frequently updated Microservices, DevOps enables dramatic acceleration of improvement.

How fast? In their 2018 State of DevOps Report, the DevOps Research and Assessment (DORA) team tell us, “…elite performers are optimizing lead times, reporting that the time from committing code to having that code successfully deployed in production is less than one hour, whereas low performers required lead times between one month and six months,” estimating that, “the elite group has 2,555 times faster change lead times than low performers.” An article in ZDNet from 2015 reported, “How Amazon handles a new software deployment every second.”

Change at the speed of business.

**Superior Business Outcomes**

Superior Business Outcomes are the very “end” goal of every process, including DevOps. Microsoft CEO Satya Nadella reminds us that today, “Every company is a software company,” drawing a direct relationship between how quickly a company can improve their software and how quickly they increase their profits. Some DevOps goals will relate to the outcomes achieved through the software continuously improved by DevOps processes.

The components of continuous improvement include velocity, quality, performance, and outcomes. Some goals may also be set in consideration of known challenges within the enterprise. Whatever goals and objectives a given organization identifies, it is critical to relate them to the value received.
What Elements of DevOps Should You Measure?

DevOps combines three key elements, people, process, and technology, to achieve the startling acceleration in software delivery it helps organizations achieve. Most DevOps metrics will correspond to these fundamental components.

People

People related metrics include task duration, response times, incidence of failure, and more. These are often the most difficult metrics to obtain so they are always the best place to start.

Process

Process is what DevOps is really all about. The process of obtaining user feedback rapidly drives the fast development and delivery of software upgrades which are deployed immediately by operations who then obtain the next round of user feedback to begin the process again. Quality and performance gains from one iteration to the next are a key metric, but a highly subjective one. Development-to-Deployment time is more objective and measurable, most useful when combined with velocity, relevance, effectiveness, efficiency, and smoothness of flow.

Technology

Technology metrics encompass hardware, software, and service functions. System uptime is critical. Software failure rate connects directly to development and deployment metrics. It’s pointless to be moving fast when the failure rate is too high.
Velocity is Key to DevOps

Velocity is the key consideration, followed closely by performance quality, though DevOps experts frequently cite their willingness to “break things” along the way and learn from those failures. Increasing competitive pressure drives an ever-increasing need to achieve continuous rapid improvement. Software developer Stackify lists useful metrics contributing to the achievement of high-speed iterations:

Deployment frequency

One objective quantity which can easily be tracked is counting the number of deployments performed by the DevOps team. The goal is always to deliver smaller improvements more often.

Change volume

How many improvements and changes derived from user feedback are embodied in each new deployment?

Deployment time

Reducing the time expended at every step of the DevOps process contributes to increases in overall speed. Clocking the actual time it takes operations to deploy new improvements helps to determine whether this task is contributing appropriately.

Lead time

Expanding beyond deployment time, lead time measures the elapsed time from receipt of a new request to availability in production.

Stackify lists
https://stackify.com/15-metrics-for-devops-success/
Customer tickets

Trouble tickets are the most available metrics of software bugs and other deficiencies that cause rework and user disruption. This is a key element of quality.

Defect escape rate

Code defects are going to happen. That’s inescapable. But you’d clearly prefer to catch those defects in the quality assurance (QA) testing stage than have users report them in production. Comparing defects caught in testing to those caught in production is a useful gauge of the efficiency of both the development process and the testing infrastructure.

Availability

Anyone in IT is vividly aware of the importance of “five nines” availability, the ability to keep the system available for users 99.999% of the time. When users complain of “constant downtime” having this ratio calculated helps to resolve their concerns while promoting deeper analysis to determine root cause of this perception.

Service level agreements (SLA)

Commitment is a key element to obtaining the confidence of the user community. Establishing a firm Service Level Agreement and regular reporting on fulfillment of it achieves this, as long as you are continually meeting or exceeding your agreements.

Failed deployments

Failed deployments may be the result of development issues, insufficient communication between development and operations, or other factors that result in incomplete integration with previous versions. While occasional deployment failures are to be expected, a pattern of failures should prompt deeper root cause analysis.
Application usage and traffic

Developers code solutions for their users to use. If your traffic monitoring reports no activity there is clearly a problem that must immediately be addressed. Similarly, if you see inordinate traffic there may be a faulty service causing the anomaly. If no error is found, significantly high traffic will be an indication of popular features in frequent use by many users. This helps to identify ways in which the product can be improved to provide greater benefit to users.

Application performance

Many elements can deteriorate application performance. The causative factor may come from the code, the storage, compilers, the database itself, protocol errors, the service bus, or many other elements. The first step in any system remediation is identification of root cause. Effective application performance monitoring is a requirement in all environments to facilitate maintenance of optimal function.

Mean time to detection (MTTD)

The DevOps “need for speed” extends beyond development and deployment to include detection of anomalies. The faster you detect them, the faster you can resolve them.

Mean time to recovery (MTTR)

The other end of the error handling sequence, once you’ve detected and identified an anomaly the time it takes to actually resolve it and return the application to full availability must be measured.
Don’t Let Perfection Be the Enemy of Good

Many complain that it’s “impossible” to measure DevOps. From an end-to-end perspective that may be so, but clearly there are many variables both objective and subjective which can be measured to provide valuable and useful insights.

Getting Started Building Your DevOps Initiative

The book The Road Less Travelled by Dr. M. Scott Peck, begins with the simple statement, “Life is difficult.”

Following that example, this text might have started with the statement, “DevOps is difficult.”

Challenges involved in the implementation of DevOps within your organization have been highlighted throughout this text. Organizational alignment challenges. Process challenges balancing the need for efficiency with the goal of delivering great software to users. Technology challenges in leveraging automation to accelerate repetitive processes and augment human effort, and the delicate balance between speed, security, and cost.

Planning to meet these challenges and overcome them to accelerate and improve the development of useful software is the first step in getting started on your DevOps initiative.
Startup Considerations

» Begin with the reality that the teams brought together in DevOps, development and operations, are not the same. Challenges to adoption of new processes, procedures, tools, and more will be different from team to team. No matter how well they align with each other, they still have different goals and responsibilities. The goal is to align these with the final goal of achieving great software with continuous improvement brings them together.

» The DevOps approach to delivering software is also radically different from what any of them are accustomed to. Delivering small improvements far more often is completely new to most.

» Some define DevOps as just the automation of repetitive processes to reduce time and expense. That’s an important component, but just one component of a much larger initiative. Others think that DevOps is simply a specific set of tools to be used when transitioning software.

» Some want to rush in and incorporate DevOps into their organization quickly. When they fail quickly, they encounter great frustration. DevOps is not something that will be adopted across any organization quickly. Full benefit may take years.

» Some fail to take a hard look at the silos of data and processes they’ve built up over the years. They fail to topple those silos and don’t achieve the change they hoped for.

» Underlying all other reasons for the difficulty in implementing DevOps is the continuing misunderstanding and misinterpretation of what it actually is. It’s far more than just a methodology, but people who think that’s what it is think it’s just a different way to build software. Many who have achieved success immediately demonstrate that DevOps is meant to be pervasive throughout an organization, and not just limited to development and operations. They may create a separate “DevOps department” and fail to convey to all users the difference and the advantages they will receive from the CI/CD approach to software deployment. They also fail to impart to users the importance of their participation in providing rapid and detailed feedback.
Don’t be daunted by finding yourself stalled by these and other difficulties. According to a DevOps.com survey, 44% of respondents who have an interest in incorporating DevOps have not yet done anything about it.

As with any new project, your DevOps initiative begins with an assessment, in this case an assessment of the current culture of your organization and how well it may align with becoming a DevOps culture. How deep in the entrenchment of each of the parties involved in their own processes and objectives? How difficult will it be to persuade them to embrace change? Just how much is yours a finger-pointing culture of blame-game?

You must also assess your organization’s real need for the DevOps strategy and philosophy. If yours is a smaller organization using software at much more fundamental levels without the frequent need for changes and improvements, DevOps may not hold much benefit for you.

This brings us to an assessment of all applications currently in use. Which ones would be more productive were they to enjoy and benefit from the iterative process of constant improvement? Which are truly monolithic blocks of code that would be difficult at best to impact in small ways frequently and rapidly? Of those, which is the organization ready to invest in re-imagining as more fine-grained, loosely coupled processes leveraging microservices, containers, and cloud transport? Would the reduction in time-to-market for various products more than offset the investment in increased velocity?

In the area of objective versus subjective assessment, also consider just how objective you and your team can be in making an objective assessment of your own flexibilities and willingness to dramatically shift your paradigms.

If your answer to that last question is less than enthusiastic, or you simply want to accelerate your progress toward full implementation of DevOps, your best strategy may be to engage a qualified, experienced DevOps consulting firm that has performed many such transitions. Their experience alone is invaluable in that it is something you can’t possibly possess yourself before you begin your own.
Some of the important stages a DevOps consultant can accelerate and improve include:

**Leadership Team Development**

Whether or not you intend to create a dedicated DevOps team, the process itself requires ongoing leadership, people who can be trusted to keep DevOps initiatives on track observing new practices and methodologies. If not a dedicated team, ambassadors from each department involved may be recruited into the leadership effort.

**Culture Transition**

DevOps adoption is more a cultural shift than anything else. The attitudes required are dramatically different, focusing on rapid introduction of improvement in successive version releases, promoting mutual feedback between operating departments, IT operations, and development resources. Culture changes don’t happen easily and must be carefully shepherd and led. Qualified DevOps experts bring the experience and skills necessary to help your leadership team cultivate and promote the new culture.

**Strategy**

In a DevOps environment, strategy is a living thing that evaluates and responds to constantly changing elements of the business environment, especially the responses of people as things change. It becomes critical to the ongoing success of the initiative to select and prioritize which projects will be handled first, second, and so on. Incorporated into strategic planning are the interactions and dependencies that exist between each project. Including the services of an external advisor avoids the pitfalls of attempting such strategic planning from scratch.
Detailed Plan

For the first iteration of any DevOps project a detailed plan will guide everyone’s actions. External experts assure that the plan is complete, comprehensive, and well-aligned with DevOps principles. Future plans will be more informed by feedback from users and customers.

Process Training

DevOps training includes Agile methodology training. External experts who’s organization is steeped in Agile methodologies can incorporate this knowledge transfer alongside DevOps principles.

DevOps Consulting Is a Logical First Step

Great DevOps engineers are hard to find, even harder to hire away from their current employment, and most likely compensated far beyond most budgets. So how does an organization enthusiastic to launch their DevOps initiative with complete and total buy-in at all levels avail themselves of the services of such a valuable resource?

Engage them.

Cloud computing has taught us that there isn’t very much that you can do yourself that you can’t have done for you by an external resource. Launching your DevOps initiative is no exception.
About Tiempo

Tiempo is widely recognized as one of the leading software engineering companies in the US. Using a combination of nearshore engineering resources, high-performance teams and relentless focus on client outcomes, Tiempo designs, builds and deploys software that makes lives better.

Tiempo is headquartered in Tempe, Arizona, with four worldclass software development facilities in Mexico. Tiempo has been recognized annually by Inc. Magazine as one of the Fastest-Growing Private Companies in America.

U.S. HEADQUARTERS
1050 W. Washington St, Suite 120 Tempe, AZ 85281 USA

GUADALAJARA
2464 Justo Sierra Guadalajara, Jalisco, México

GUADALAJARA
San Dionisio 25 Jardines de San Ignacio Zapopan, Jalisco, México

HERMOSILLO
545 García Morales Blvd. Hermosillo, Sonora, México

MONTERREY
CIT2 Eugenio Garza SadaMonterrey, Nuevo León, México

www.tiempodev.com

Contact us at:
contact@tiempodev.com
602-910-4646