The Way of DevOps: A Primer on DevOps Principles and Practices

What Is DevOps?

DevOps refers to the combination of software development (which includes software testing) and operations methodology (the values, principles, methods, and practices) to deliver applications and services. DevOps promotes frequent communication and ongoing, real-time collaboration between traditionally disparate workflows of developers and IT operations teams. The DevOps approach to organizing workflows replaces siloed development and IT operations teams with multidisciplinary teams capable of implementing Agile planning and practices, such as continuous integration, continuous delivery, and infrastructure automation.

The technology community uses a variety of terms to describe the essence of DevOps. It is a culture, a movement, a philosophy, a set of practices, and the act of using tools (software) to automate and improve imperfect methods of managing complex systems. Regardless of how you characterize it, DevOps is best defined by the *why*: DevOps exists because software development is a strategic asset for organizations and businesses of all sizes in the modern global economy. Therefore, DevOps practitioners seek to create, operate, and manage complex systems with a culture of continuous improvement and a sense of urgency to meet the demands of a software-for-profit model.

The DevOps culture is reinforced by the practices it borrows from Agile and Lean principles, with an added focus on service and quality. By designing, building, testing, deploying, managing, and operating applications and systems faster and more reliably, DevOps practitioners seek to create value for customers (a profitable competitive advantage) and foster a manageable workflow that places people over product.


elopment

A History of DevOps

Advances in computing and the economic momentum of the U.S. technology sector in the early 21st century provided companies (that had never connected existing IT operations as a strategic asset) with a new competitive advantage. In previous economic expansions, technological innovations were reserved for manufacturing core business products and developing services to support customers. Internal human capital, technology, and advanced software and systems kept the infrastructure stable and the lights on for sales and marketing. Companies like Google, Netflix, and Amazon demonstrated how to create value by leveraging internal software development and IT operations to innovate and rapidly respond to customer's increasing demand for reliable, secure, and feature-rich applications. To do so, companies' traditional principles and practices of software development and operations had to adapt to a new reality: They had to market working software to users at faster speeds while improving service levels and security.

Various philosophies, like Agile development, grew out of the recognition that the old way wasn't enough. The growth of Agile principles of project/product management, combined with advances in computing technology (like cloud-based infrastructure), highlighted the conflicts between development and IT operations teams: Their opposing goals made it impossible to frequently deploy reliable and secure systems. To compound these issues, technology professionals were working long hours, weekends, and holidays, and experienced a reduced quality of life. Caught up in the market demand for more, faster, and better, these pros knew they needed different solution. Enter DevOps.

The DevOps movement is an effort to transform the way various stakeholders interact, communicate, and work together in the software development lifecycle. John Willis, former Director of Ecosystem Development at Docker and author of six books on enterprise system management, points out that the reason people disregard DevOps as something new or worthy of attention is the lack of a "canonical definition." Unlike for Agile, there is no "DevOps manifesto" to bind the movement to a specific birthplace and time or guide those looking for a specific set of values, methods, practices, and tools.

The disregard for the movement's formal establishment (i.e. the "we've been doing DevOps for years" mentality) stems from the fact that DevOps reflects familiar business process principles created in the 20th century. It is a combination of philosophical management and manufacturing movements. Lean principles (Lean Startup), the Toyota Production Management (Toyota Kata), and Agile software development (continuous integration/continuous delivery, software test automation, etc.), merged with ideas that were spreading through Agile, Velocity, and DevOpsDays conferences.

The term DevOps is attributed to Patrick Debois, Andrew Shafer, and their work on Agile Infrastructure at a 2008 Agile conference. DevOps gained momentum in 2009 at the Velocity conference during a presentation by John Allspaw and Paul Hammond: "10+ Deploys per Day: Dev and Ops Cooperation at Flickr." Debois believed DevOps was a response to inflexibility and the "us versus them" mentality that organizations created by siloing software developers, testers, managers, DBAs, network technicians, system administrators, etc. Technology helped create a harmonic convergence of DevOps management principles and software tools capable of introducing critical DevOps practices, like collaboration, automation, monitoring, logging, and the deployment of software. Using a combination of software tools and a culture of collaboration and constant improvement, the DevOps movement for software development was born.

DevOps: Fact or Fiction

There is a common trend among the growing community of DevOps thought leaders, proponents, and practitioners to dispel the myths about DevOps. Determining what DevOps is and is not is part of the ongoing dialogue in the tech community that the method is designed to impact. This dynamic is common in the technology sphere; for example, it's similar to the way Agile software development gained use and acceptance.

The first novel about DevOps, The Phoenix Project, is a fictional account of a company struggling to keep up with competitors and manage a failing IT organization. It contributed to the growth of DevOps and interest in the methods early adopters introduced in 2013. Bill, the novel's protagonist, endures his journey as an IT manager trying to solve the kind of problems readers might recognize in their own organization's efforts to develop software and manage technology operations in the age of rapid releases and continuous change. As a coda to the novel, the authors — anticipating misunderstandings about DevOps — present common myths that mischaracterize the DevOps movement.

Separating fact from fiction and removing misconceptions about DevOps is an important step in understanding how the concept helps transform software development and IT operations. Remember, DevOps is not any of the following:

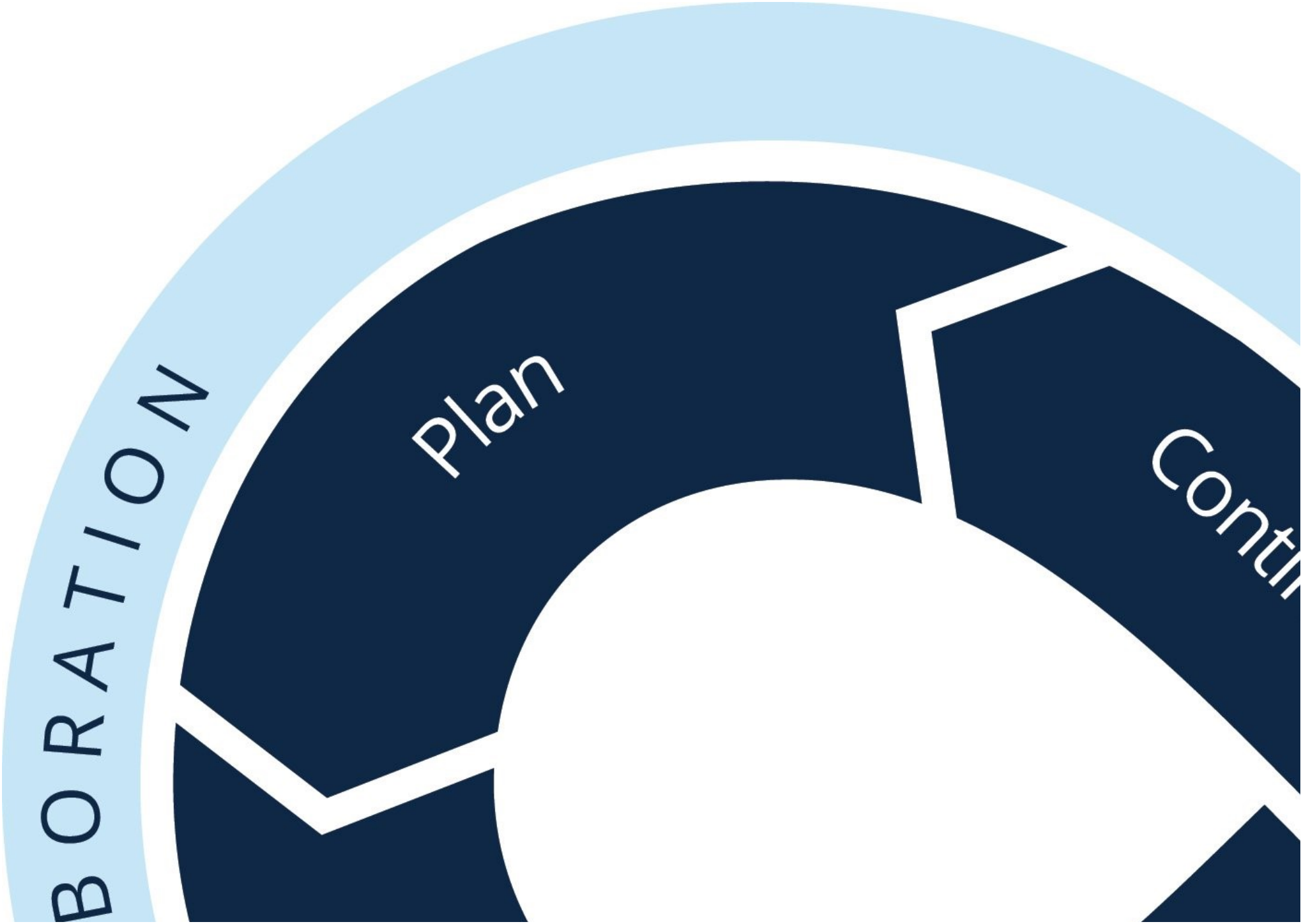
- A replacement for Agile
- A replacement for manual testing and QA teams
- A replacement for operations
- Dependent on open-source technology
- A software tool or service

DevOps IT: Toolchain and Architecture

While *DevOps-as-a-Service* and *DevOps products* are marketing concepts designed to communicate with a specific audience, there are particular software tools designed to empower DevOps practices and processes. We sometimes refer to the technology stack of software tools you use in DevOps environments as a toolchain. Adopting a DevOps model of software development relies on effective *tooling*, i.e., selecting the right technology to manage complex systems and scale workflows in fast-paced, high-pressure environments. The DevOps toolchain is a

categorization method to distinguish different tools you use at different stages of software development, testing, and IT operations business processes. These categories include the following:

- Develop code
- Build with continuous integration
- Test automation
- Package application pre-deployment
- Manage releases
- Configure infrastructure
- Monitor performance


BORATION
Plan
Conti

Microservices, Virtualization, and the Cloud

Effectively implementing DevOps tools and practices to test, deploy, monitor, and modify the complex code and systems behind the economic engines of Netflix, Amazon, Microsoft, and Google often requires a software architecture known as *microservices*. Sam Guckenheimer of Microsoft Visual Studio describes microservices as "the architectural pattern of composing a distributed application from separately deployable services that perform specific business functions and communicate over web interfaces." Guckenheimer, author of four books on DevOps and Agile practices, recommends using microservice architecture in combination with DevOps practices to shorten the time it takes to deploy changes in code or configuration.

Working with small, reusable building blocks of code ensures that the application under development is not affected by the increase in the velocity of deployments in the DevOps environment. When implementing cloud-based microservice architecture at scale, tech leads often use *containers* to isolate, package, and deploy code.

Containers are the next evolutionary step in virtualization technology. *Virtualization* refers to the act of using a virtual resource (for example, a server, desktop, operating system, or network) to make computing processes scalable. Working in place of hardware, a *virtual machine* is the data on a computer that uses a file management system that enables you to make changes more efficiently. According to Guckenheimer, containers are more "lightweight" than virtual machines and provide a simplified method for configuring the file management that empowers DevOps. The cloud-based infrastructure empowers microservice architecture. Critical DevOps practices, like monitoring and logging code changes in high-velocity virtual production environments, require scalable, secure, and stable cloud-hosting platforms.

Agile-Inspired DevOps Methodologies

DevOps has extended the principles of Agile software development beyond the deployment of product (software code) to include the services provided by IT operations after the product deploys. In other words, DevOps picks up where Agile left off and incorporates a focus on overall service once product is delivered to the customer. The practice combines development and operations — from the design and development stages of the Agile SDLC all the way to production support in IT operations. Like Agile, DevOps is a concept that eludes simple definitions. To better capture the essence of this broad methodology, you need to experience it in levels using the same framework the *Agile Manifesto* provided.

DevOps Values

Without the appropriate workplace culture, you cannot build the best products with the best technology and offer the best service. This rule applies to a DevOps environment as much as it does to any environment where people manage complex systems. DevOps is unique to software development methodology because its practice promotes empathy via a culture of collaboration and communication, rather than encouraging siloed functionality. Leading a change management effort to organize with DevOps is not easy or quick. Amazon made a consistent and clear long-term investment in leveraging DevOps methodology as the company continued to grow, learn, and overcome the common obstacles associated with large-scale change management and infrastructure design.

John Willis created the CAMS model (culture, automation, measurement, and sharing) to describe his thoughts on the values that guide DevOps:

- **Culture:** DevOps seeks to solve business problems that arise when people create and manage complex systems. In this regard, DevOps is as much a method for managing human problems as it is a technological solution. A "people over process over tools" culture is a central tenet of DevOps. Even with the advent of more innovative tools and advanced computing technology, the process of software development depends on elements of human culture. Its practitioners must create an environment of open communication, in which the mutual goals of and understanding between all stakeholders drive production. Teams form around mutual product- or project-oriented outcomes and avoid prioritizing individual skillsets or siloing functions.
- **Automation:** DevOps is not just about tools or about automating tasks using software. That said, automation is a core DevOps value, and it is essential to leveraging Agile development practices, including continuous integration and continuous delivery. To accommodate continuous releases, DevOps encourages automation. In the DevOps methodology, it's critical to prioritize problem solving that uses automation, and to make QA everyone's responsibility. Embracing the idea that something will go wrong and acknowledging the inherent instability in complex systems are the purposes of automating tests and empowering a rapid response. DevOps focuses on continuously employing automation to create stress and stimulate partial failures of complex distributed systems in order to drive improvement and speed up production. For more on software test automation, check out this article."
- **Measurement:** In order to determine if DevOps is continuously improving processes, team members should collect and analyze data. This mandate applies to business-side metrics as well as development, test, and operations metrics. One of the basic metrics to consider includes *mean time to recovery (MTTR)*, which is the amount of time it takes to recover from a product or system failure. There are numerous other considerations as well: How long does it take information to flow to the right people? Is it the right information to avoid bottlenecks? How long does it take to move through the entire development-to-deployment lifecycle? How often are recurring problems popping up? What is the current state of employee satisfaction? Measuring and incentivizing the right behavior is a core value of DevOps. Valuing measurement and verifiable behavior emphasizes learning and consistent improvement — two essential aspects of DevOps culture.
- **Sharing:** Willis referred to this value as the *loopback* in the DevOps cycle, where stakeholders share ideas and solve problems. Sharing ideas helps attract talented people who thrive on feedback to improve. DevOps depends on the principles of continuous improvement and the collaboration those principles foster. Sharing is a core value of DevOps, as development and operation staff share traditionally separate team functions. In DevOps, teams embrace the idea that everyone is involved in building and shipping an application and making sure it meets customer and business goals. Sharing means being transparent and valuing the findings and data from all teams. The IT operations' service desk is often responsible for operational reporting. DevOps encourages this position to openly share data with all teams in the value chain.

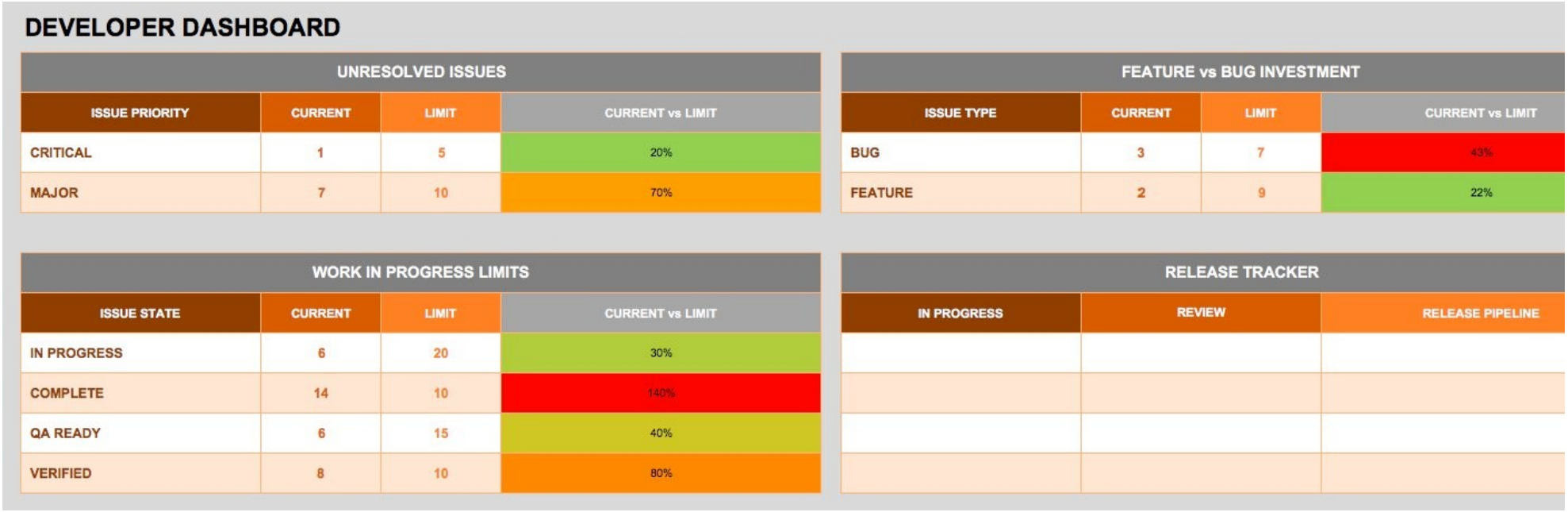
DevOps Principles

The Phoenix Project introduced "the three ways" as the core principles that guide DevOps methodology.

- **The First Way:** Think in terms of the overall outcomes of a system. This means that you should approach the flow of work (from development to operations to the customer) in "small batches" while optimizing the comprehensive, global performance of a system. This principle de-emphasizes localized goals and individual performance metrics (for example, productivity goals specific to development). With systems thinking, you don't pass defects (bugs in the code) downstream (or left to right, in the case of Agile visualizations) in the workflow. The first way requires continuous build, integration, and deployment principles in on-demand environments. The focus here is to limit work in process and keep change constant and safe.
- **The Second Way:** Amplify the constant right-to-left *feedback loops* in the designed development workflow to prevent problems and enable faster detection and resolution. Use short, effective feedback loops to create process, design workflows, and visualize data. The constant flow of feedback from the opposite direction of the designed workflow (shifting left in the Agile *value stream*) helps prevent, detect, and recover from inevitable failures and bugs. Amplifying feedback loops creates "quality at the source." This DevOps principle includes the necessary and disruptive practice of stopping production when there are deployment failures. The second way emphasizes daily improvement over daily work, and also champions automation whenever it is available to ensure that code is always optimized. Most importantly, it encourages all stakeholders to share the collective pain.
- **The Third Way:** Create a culture that embraces experimentation and understands that daily practice and repetition lead to mastery. This principle embodies the "fail fast" mentality popular among the titans of tech (including Facebook and Google). A culture of *doing* instead of over-analyzing enforces a learning environment where success and failure occur at regular intervals. Failure and problem solving lead to more secure, reliable, and innovative systems. The third way requires a high-trust leadership environment that reinforces improvements through risk-taking.

DevOps Dashboard Template

In DevOps, it's critical to create an environment where everyone can see when and where the work in progress is not achieving business and customer goals. Using dashboards that promote open communication and collaboration, you can measure and share critical data to keep feedback loops amplified and projects on track.



DEVELOPER DASHBOARD

UNRESOLVED ISSUES

ISSUE PRIORITY	CURRENT	LIMIT	CURRENT vs LIMIT
CRITICAL	1	5	20%
MAJOR	7	10	70%

FEATURE vs BUG INVESTMENT

ISSUE TYPE	CURRENT	LIMIT	CURRENT vs LIMIT
BUG	3	7	43%
FEATURE	2	9	22%

WORK IN PROGRESS LIMITS

ISSUE STATE	CURRENT	LIMIT	CURRENT vs LIMIT
IN PROGRESS	6	20	30%
COMPLETE	14	10	140%
QA READY	6	15	40%
VERIFIED	8	10	80%

RELEASE TRACKER

IN PROGRESS	REVIEW	RELEASE PIPELINE

Download Development Operations (DevOps) Dashboard Template

DevOps Methods

The *Agile Manifesto* upended traditional project management and software development processes. DevOps does not try to replace Agile (because it is compatible with the principles of Agile); instead, the DevOps methodology identifies areas in which Agile has room for improvement. DevOps' primary focus moves beyond product release to incorporate and emphasize the service side of operations in the overall software-for-profit model. DevOps provides coverage for IT operations teams by not leaving them to reinvent the Waterfall method of software development.

Like Agile, DevOps borrows from Lean Management, Scrum, Extreme Programming, and Kanban methods to work in sprints and manage the capacity of different functions to help teams adapt to business and customer needs. The DevOps methodology integrates operations with the development, QA, and product management principles of these prior methods to emphasize the following actions: making continual improvements to working software and small batches of code; limiting work in progress bottlenecks; and amplifying feedback. A disciplined DevOps approach enables rapid Agile development with a focus on production-ready code, repeatable processes (automation), and synchronized operations tasks that minimize backlogs.

DevOps Practices

The DevOps methodology leverages key practices and techniques to streamline software development and operations processes. Increasing the frequency and number of daily software deployments is challenging to teams of all sizes with varying access to resources. To manage the organizational challenges of DevOps, its practitioners leverage practices known as continuous integration (CI) and continuous delivery (CD).

- **Continuous Integration:** CI is the practice of engaging in ongoing testing (leveraging automation) by merging code development with real-time, problem-seeking tests. The goals of CI are to reduce integration problems, improve quality, reduce time to release, and empower feedback loops that contribute to higher-velocity (daily) deployments. CI leverages comprehensive, automated testing frameworks and continuously addresses problems to keep the system in a working state.
- **Continuous Delivery:** CD is the practice of frequently building, testing, and releasing code changes to an (production or testing) environment in small batches after the build stage. The emphasis on automated testing (and automated builds) for quality assurance capitalizes on the efficiency of successful test automation and is essential to the deployment-ready goal of this practice. You can achieve CD when you synchronize it with the steps required for CI. However, CD does not require the deployment of every build. Continuous deployment deploys every CI build to production.

Continuous delivery was created by Jez Humble and David Farley. Humble emphasizes that while CD is widely used by web companies, the practice's techniques apply to a variety of industries, including any company that considers software development capability to be a strategic asset.

DevOps Glossary of Terms

Below, you'll find a list of essential DevOps terms:

- **DataOps:** This is an approach to data analytics (data operations). DataOps is a process-oriented method that borrows from Agile development, Lean management, and DevOps. It seeks to merge statistical processes of monitoring and controlling with these other methods to improve the throughput velocity, quality, security, and reliability of data analytics at scale.
- **Site Reliability Engineering (SRE):** Founded by Ben Treynor Sloss, VP Engineering at Google, SRE is "what you get when you treat operations as if it's a software problem." Google SRE is a team that uses software as the primary tool to "manage, maintain, and mind" systems and is also a set of principles and work practices. SRE refers to an environment in which a lead might ask a software engineer to design an operations function in an area (e.g., production) devoted to engineering work. SRE uses "source-level access and the moral authority required to fix, extend, and scale code to keep it working, harden it against the vagaries of the internet, and develop our own planet-scale platforms."
- **Systems Administration (SA):** A systems administrator is responsible for managing, configuring, operating, and servicing the operations of systems in a computing environment (for example, in a network of computers or servers). An SA solves problems related to computing systems, including troubleshooting applications, managing systems users, implementing security policy, and performing user training and systems-related project management.
- **Infrastructure as Code:** This is the practice described by SRE (see above). In this process, you manage software development infrastructure (systems) programmatically, using code and techniques (like continuous integration) and optimized and repeatable methods to speed up deployment. Systems administrators interact with systems infrastructure, often in a cloud-hosted environment, using tools (software code) and automation instead of manual configuration.
- **Configuration Management (CM):** This is the practice of managing change systematically over time to maintain system integrity. In software development, the CM process identifies system changes by attributes at specific points in time to manage change over the course of the development lifecycle. In DevOps, this is managed with automation to make the process repeatable and reliable at scale for increased velocity requirements.
- **Cycle Time:** The time it takes to design, build, deploy, and monitor working software delivered to users after observation, experimentation, and analysis of data-informed business cases and results. Shortened cycle times result from DevOps practices such as continuous delivery, automation, and microservice architecture design.
- **Validated Learning:** Using actionable data from performance metrics and amplified feedback loops involving all stakeholders, at every stage, to drive continuous improvement.
- **Version Control:** Monitoring changes using tools like GitHub to communicate during daily software development and operations activity and integrate with other software tools.
- **SAFe Framework:** Scaled Agile Framework (SAFe®) is a scalable, modular approach for implementing Agile in a way that best meets the needs of the organization. It is available as an online knowledge base for implementing Lean-Agile development. SAFe provides "comprehensive guidance for work at the Portfolio, Large Solution, Program, and Team Levels."

Adopting the DevOps Model

Adopting DevOps requires sound change management principles and organizational support of leadership and all stakeholders involved in developing software as an asset. DevOps embraces Agile development processes and borrows from other methods that emphasize increased production, systems infrastructure, automation and configuration management principles, and other development practices. DevOps is a shared movement of supporting knowledge from various influencers that represents various expertise and fields of practice. While not true for all software tools adopted into the DevOps methodology, the open source community is well represented in DevOps.

Building the DevOps leadership culture is vital in order to realize the desired outcomes of streamlined code and stable systems. Forming cohesive, cross-functional teams comprised of development, QA, and IT operations professionals requires a culture of collaboration. It is important to measure data and focus on metrics that provide insight to the DevOps cultural shift. The annual State of DevOps Report from Puppet and DORA provides results of surveys designed to measure the impact of organizational culture on the state of DevOps. The 2017 report measured the impact that leadership characteristics like vision, inspirational communication, intellectual stimulation, support, and recognition have on transforming organizations to high-performance DevOps. The findings indicate that advancing the progress of DevOps adoption requires the following:

- Transformational leadership
- Focus on automated configuration management, testing, deployments, and change approval processes
- Faster throughput of code and more stable systems through emphasis on quality builds
- Developing “loosely coupled” architecture where teams release and make changes independent of one another with continuous delivery
- Lean product management practices that create fast cycle times and promote experimentation and feedback

If you are in the advanced stages of implementing DevOps methods, measuring your progress and identifying areas of focus is critical to improving. Jez Humble, one of the authors of the State of DevOps Report and co-creator of continuous delivery practices, recommends a set of questions to determine the status of DevOps adoption:

- Is every stakeholder checking in to their shared environments daily?
- Are tests run at every check in?
- When the build breaks, how long does it take to fix?

These insights help to determine if your team is applying DevOps methods correctly, and along with survey questions designed to measure the state of your team’s DevOps culture, can help identify where you’re falling short of DevOps adoption.

- Is information actively sought out on your team?
- Are failures treated like learning opportunities on your team without punishment?
- Is there a culture of shared responsibility on your team?
- Is collaboration of different functional teams encouraged and rewarded?
- Are new ideas welcomed or is there a fear of experimentation?

Why DevOps Matters

The State of DevOps Report (2017) found that high performing DevOps teams increased the frequency of code deployments, reduced the lead time to deploy, increased time to recover from system failure, and lowered the change failure rate. Companies that rely on software development as an asset, or exist to sell software for profit, must continuously find ways to create value for their customers and competitive advantages for their business processes. DevOps aims to reduce development cycles, increase deployment frequency, and create stable systems in alignment with business objectives.

In addition to the operational benefits of enhanced throughput and stability of code, DevOps strives to balance the high-pressure, on-demand environments of software development and operations professions. DevOps methods focus on reducing unplanned work and collaborating openly to improve individual and team relationships. These values foster an environment that creates faster time to market for product teams with lower failure rates. In response to stressful events during product development, DevOps culture advocates for learning driven by overcoming failure and accountability for the whole system, regardless of specific functional contributions by individuals or product teams.

The DevOps Profession

There is confusion and varying opinions in the DevOps community about using the term to describe a profession. The argument is that the job title is redundant, that engineers and operations professionals apply DevOps methods to design, develop, test, manage, and deploy software. That doesn’t make this professional a “DevOps Engineer.” The important distinction is that DevOps methodology is available to companies and teams to implement without specialists wielding the title or previous experience applying its principles and practices.

However, if a team prefers to work with DevOps professionals of merit who leverage experience, knowledge, and knack for implementing DevOps principles and practices on successful projects, there is an active market of in-demand professionals available. Keep in mind that teams composed of “DevOps Engineers” leveraging “DevOps tools” and automation can stray from the principles of DevOps if a culture of collaboration and continuous improvement is ignored.

DevOps FAQ

Determining if DevOps is a better way to improve performance and achieve a culture of collaboration leads to some common questions that add perspective on the foundations of the methodology.

- **What Problems Does DevOps Solve?** It is increasingly more difficult to make organizational changes and business decisions in the modern economy without some change to IT systems and operations. DevOps offers a methodology to address the inevitable failures and the historical bottlenecks experienced by development and IT operations who manage change separately and with different functional goals.
- **What Is the Goal of DevOps?** DevOps is an organizational culture that leverages core principles and technical practices (for example continuous delivery) to optimize application performance and infrastructure and deliver value as a strategic asset. According to the 2017 State of DevOps Report, high-performing DevOps teams achieved this goal as measured by:
    - 46 times more frequent code deployments
    - 440 times faster lead time from commit to deploy
    - 96 times faster mean time to recover from downtime
    - 5 times lower change failure rate (changes are 20 percent as likely to fail)
- **What Tools Are Used in DevOps?** Implementing and mastering core DevOps technical practices like continuous delivery is more important than the specific tools used. However, there is a robust market of tools and technology that DevOps practitioners use to achieve faster throughput of code and create and maintain more stable systems. These include, but are not limited to the following:
    - Code repository
    - Build servers
    - Configuration management tools
    - Virtual infrastructure (hybrid or private)
    - Test automation

The State of DevOps Report concludes that teams handcuffed by commercial off-the-shelf software (COTS) can apply continuous delivery to any system as long as it is “architected correctly.” For more on DevOps tools, see “DevOps Tools Smartsheet…”

Improve DevOps with Smartsheet

Effective communication and collaboration between operations and IT departments and a culture of continuous improvement can drive success in any business. That’s why creating a DevOps culture is essential to your company’s survival, and introducing tools that support and encourage collaboration is an important first step. Smartsheet is a work management and automation platform that enables enterprises and teams to work better. Over 65,000 brands and millions of information workers trust Smartsheet to help them align the right people, resources, and processes to get work done. The world’s leading IT professionals rely on Smartsheet to help increase throughput and operate at maximum efficiency.

Use Smartsheet to improve accuracy with real-time plans, increase collaboration with internal and external teams, and boost efficiency with resources centralized in one location. Quickly resolve reported issues, gain visibility into issue patterns, and maintain auditable records without additional work.

Discover how Smartsheet can help maximize your DevOps efforts today.

Try Smartsheet for Free