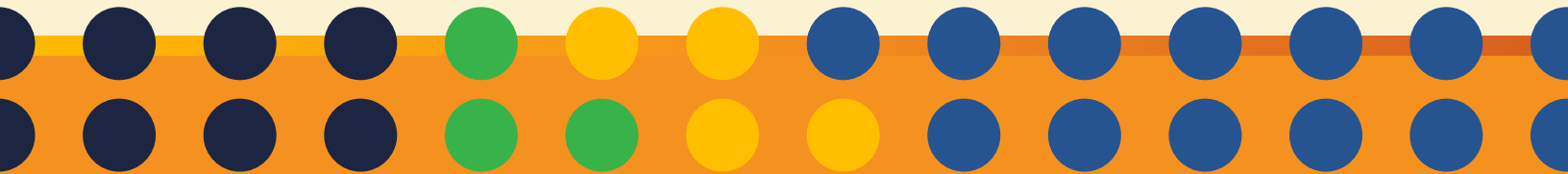


EMBRACING DEVOPS CULTURE



BY DEVELOPERS • FOR DEVELOPERS

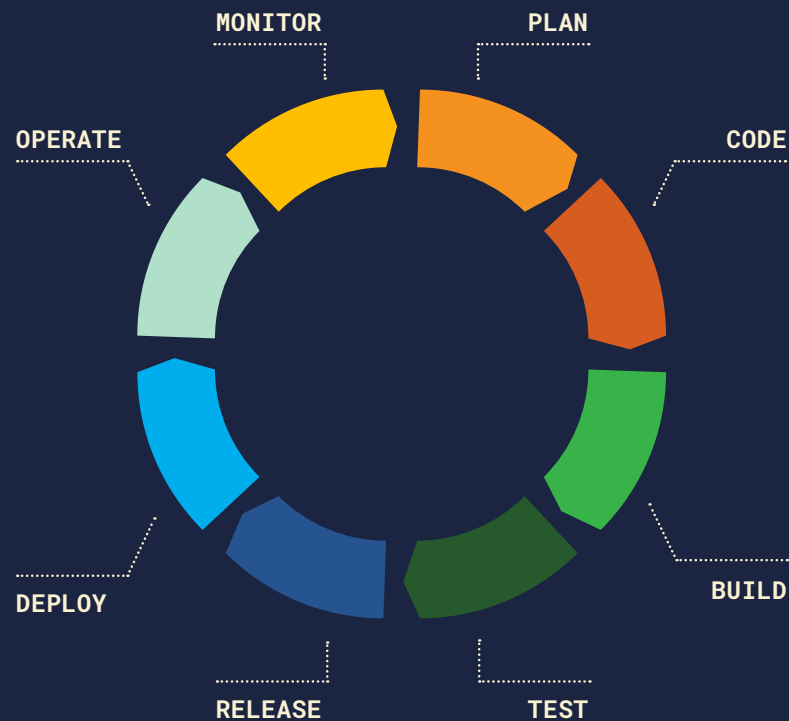
SIGNS YOUR ORGANIZATION COULD BENEFIT FROM DEVOPS:

The development team finished the product 3 weeks ago, but it's still not in production. You released a new feature and half the website went down. A significant amount of your day is spent dealing with production outages or issues. No matter what your role is, if you work in a technology organization, chances are you've heard the term DevOps. If any of the above problems sound familiar, chances are your company could benefit from adopting DevOps practices.

Before we dive deeper into other common problems that DevOps tries to solve, let's start with a brief history of what DevOps is and where it originated. A bit later we will take a more in depth look at the understanding of DevOps philosophies, and explore the key pillars of DevOps. For now, we will focus on the "what", as in what common problems does DevOps aim to solve.

WHAT IS DEVOPS?

DevOps is a philosophy that arose circa 2009 following the success of other frameworks for optimizing software delivery such as agile, scrum, and xtreme. All of these frameworks have a similar goal - develop higher quality software faster. While the aforementioned frameworks concentrate mostly on the development process, DevOps takes a more holistic approach by applying its philosophies to the entire value stream. This includes everything from the development process, to QA, to production releases and beyond. Let's explore some of the leading indicators your organization could benefit from DevOps.





COMMON PROBLEMS DEVOPS AIMS TO SOLVE

REALIZING VALUE-PROVIDING FEATURES TAKE TOO LONG

This problem can take many forms, but can be summed up by saying it takes too long for new value-providing features to make their way to production. To paraphrase, there is too much lead time in your processes and you'd like to reduce your time to market. Common scenarios that impact lead time include:

- **Unplanned Work.** The agile software development process is well defined and works great when things run smoothly. However, it seems that a large portion of your team's time is spent dealing with unplanned work. This work may come in the form of production issues, tech debt, or inefficiencies in the deployment pipeline.
- **Red Tape.** If it takes more than a couple minutes for a developer's code to make its way to production, chances are you need to evaluate your deployment pipeline. Overburdened change management processes riddled with multiple handoffs and gatekeepers that take days, weeks, or even months are a key indication that something has gone wrong. These handoffs usually involve significant manual Quality Assurance and User Acceptance testing. If bugs are found during the process, developers are asked to change context to revisit a problem they haven't worked on in days, weeks, or months. This leads to unplanned work and is detrimental to productivity.
- **Lacking Access to Production-like Environments.** We've all been part of conversations where code fails to work as expected during deployment and the developer casually mentions that it works on their machine. If a developer does not have the ability to test their code in a production like environment early and often, these types of delays will be abundant when it comes time to release your new software or feature. This leads back to Unplanned Work as the development team must change contexts again to solve the issue that is currently delaying release. Ideally production like environments can be created on demand in minutes. Unfortunately many organizations still require weeks of lead time to create adequate test environments for development teams.

ABSENCE OF PRODUCTION METRICS

One strong indicator that you have an absence of production metrics is that users tell you about software problems. If help desk tickets are the norm for detecting production issues, your software is most likely frustrating users on a daily basis. In today's world, users expect things to just work when it comes to software and the internet, and many users will not take the time to send you a help desk ticket when



software doesn't work. There are many cases of organizations missing out on tens of thousands of dollars in revenue due to a bug they didn't know about. Mature organizations have found ways to monitor users in production and detect anomalous behavior before the user reports it. This allows the company to proactively fix problems before many users are even aware there is an issue. Even if you aren't able to fix the problem immediately, you can signal to users that a fix is in progress.

Tracking errors in production is important but is only 1 of the 4 golden signals. Other important metrics include throughput, latency, and saturation of your applications. Armed with information from these metrics you can make critical decisions about the health of your production system thus improving the experience for the end user. For example, latency issues means everything the user does takes too long which leads to an unsatisfactory experience. Measuring throughout can help you understand how many users are using your application at any given time and whether your infrastructure can support it. Finally, saturation is a measure of how much of your infrastructure is being utilized; this is important for optimizing the costs within your organization.


DEPLOYING TO PRODUCTION IS SCARY

It's that time of the week / month / quarter / year again when you deploy the new features your development team has been diligently working on for the past several sprints. You've calculated when there is the least amount of traffic on your website to minimize risk. All the relevant parties are in the room, the operations engineers, QA, developers, and product managers. There is excitement in the air, but also quite a bit of apprehension. These production deployments don't usually go well and often spiral into hours of overtime and late nights. The operations team starts the deployment process and within a few minutes you begin to see the warning signs that things aren't going well. Help desk tickets are starting to trickle in as things break.

Unfortunately there is no way to reverse this process once it has started so the only path is forward. You settle in for the long haul where tempers will run short and emotions high as everyone points the finger at each other. This is a common scenario in organizations that have not adopted DevOps principles; many of these organizations don't recognize there is a better way.

INABILITY TO SCALE TO DEMAND

Companies often have peak traffic times for their applications. For a ticket sales website, this is when a new event goes on sale. For retail companies, this takes place over the holidays and may peak on Black Friday. Many organizations spend a lot of time and money preparing for these peak times. In a DevOps environment, scaling applications to handle a larger workload can be as easy as pressing a button. In sophisticated organizations that run on public or private clouds, scaling can often happen autonomously based on traffic. Automating these types of tasks frees up valuable resources to work on business value projects instead of maintaining infrastructure.



If your organization is suffering from any of the above, it is likely you could benefit from the implementation of DevOps practices. DevOps can significantly improve your engineering team's ability to adapt to market changes and release the features that give you an edge against your competition. Adopting a DevOps culture improves company morale and stops the blame game when it comes to effectively releasing software.

Now that we understand the types of problems DevOps aims to address, let's take a closer look at where DevOps originated and how its benefits can be realized in your organization

THE ORIGIN AND KEY PRINCIPLES OF DEVOPS

THE ORIGIN OF DEVOPS

DevOps as a term originated in 2009 following a talk at the O'Reilly Velocity Conference titled "10+ Deploys per Day: Dev and Ops Cooperation at Flickr." John Allspaw and Paul Hammond walked through some of the pains in the current software development lifecycle, identifying familiar contentious scenarios that had become all too common between development and operations teams. "It's not our machines, it's the developers code!". "We can't test our code because operations can't get us a production environment!" John and Paul made the case that the only rational way forward is to integrate development and operations into a more cohesive unit.

This talk is widely accepted as the birth of the term DevOps and the beginning of a movement in IT that is still very much with us today. For a more complete DevOps history checkout [The Origin of DevOps: What's in a Name?](#)

Even though DevOps was coined as a term over 10 years ago, it has become a confusing buzz word in recent years and there are a lot of misconceptions about how DevOps is actually defined. There are job postings for DevOps Engineers, a seemingly infinite number of tools in the space, and even companies offering DevOps as a service. With all of the misinformation out there it can be hard to understand DevOps. In fact, it may be easier to start with some common misconceptions before moving onto exploring a more formal definition of DevOps.

WHAT DEVOPS IS:

- A way of doing work. DevOps is a philosophy that permeates the entire organization from top to bottom. Often the journey to DevOps starts with pockets of highly motivated people or teams adopting DevOps principles. These teams are rewarded by bringing great value to the organization more quickly and reliably than other teams. At this point the benefits of DevOps are measurable and other teams in the organization begin to take notice. This new way of doing work then spreads like a virus to 'infect' the entire company.
- Solves the core IT conflict of stability vs. innovation. In traditional organizations, there is a constant struggle between those that want to add new features to software to provide more value to customers, and those that want to keep things running smoothly so as not to disrupt the stability that customers have come to rely on. DevOps aims to remove this conflict by allowing the organization to move quickly while also maintaining stability.
- A combination of lessons learned from a variety of process improvement movements. These include methodologies such as The Toyota Way, Lean Manufacturing, The Goal, Lean Six Sigma, and Agile. Each of these movements has revolutionized their respective industries and become gold standards for operational excellence. DevOps has combined these methodologies and extracted the core principles into what we call The Three Ways.

The Three Ways are instrumental to any DevOps practitioner and deserve a deeper dive. Let's break down the three ways and provide some examples of each one.

WHAT DEVOPS IS NOT:

- A rebranding of System Administrators and/or Operations Teams. DevOps requires more than a title change to be effective. It requires a shift in thinking. Many DevOps professionals become frustrated when they think they are joining an organization that subscribes to a DevOps philosophy, only to find out they are expected to do systems administration work.
- A separate team or role within the organization. While it is possible to start your DevOps journey with a single team, completing the transformation requires a cultural shift that is company wide. Companies that are hiring for a DevOps role are often either very early in their DevOps journey, or do not completely understand the movement.
- Beholden to a particular toolchain. While many professionals that subscribe to DevOps philosophies are versed in technologies for cloud, configuration management, and automation, they know that these technologies are just tools that can easily be exchanged with one another.
- Combining Developers and Operations into a single team and crossing your fingers. While DevOps advocates decreasing the void between developers and operations, simply putting them on the same team will not achieve the desired outcome if the two disciplines do not know how to work together.

DevOps is a young movement and many organizations are still wrestling with its definition and the value it provides. Consequently the mistakes above are not uncommon in this field. Let's take a look at what DevOps actually is, and how the community defines itself.

While this article provides a good overview of DevOps, I highly encourage anyone looking to continue their journey in this space to read *The Phoenix Project*, *The DevOps Handbook*, and *Accelerate*. I consider these books must reads for any new hire at Callibrity.



THE THREE WAYS

THE FIRST WAY: FLOW


The First Way says we need to accelerate the flow of work through the organization. This is often done by visualizing how work is done using a process like value stream mapping. Value stream mapping involves creating a visual representation of how work flows through an organization from beginning to end. In software we often think of the start being ideation of a new feature, and the end being the new feature running in production and available to customers. When creating a value stream map you should involve all parties that take part or have a stake in the work being done. This helps to ensure everyone has the same base level understanding and no critical parts of the work are missed.

The First Way promotes the idea of Systems Thinking, meaning we should try to look at an entire system when considering solutions, not just an individual part. It is not valuable to spend an excessive amount of time perfecting the development process if things grind to a halt once we try to deploy to production. In this case the deployment process is a bottleneck within our value stream. Once a value stream map has been completed it becomes very easy to identify such bottlenecks in an organization. A bottleneck is not always time due to time spent doing work. Sometimes the bottleneck may be due to the amount of time pending work spends sitting in a queue after a handoff, or waiting for approval in an overburdened change management process.

Another way to improve the flow of work is to automate whenever possible and decrease Work In Progress (WIP) to manageable levels. Organizations can see great productivity gains by automating tedious repetitive tasks like building applications, deploying code, and provisioning infrastructure. Decreasing WIP allows teams to focus on doing a smaller number of tasks at a time. This increased focus leads to better quality work, decreasing the likelihood that a piece of work is sent backwards in the pipeline. Ideally, all work flows from left to right and is never sent backward. Anytime work is sent back in the pipeline, someone must context switch and stop the flow of any work that is currently in progress.

THE FIRST WAY DEFINES 4 TYPES OF WORK WITHIN AN ORGANIZATION.

1. Business Projects - These are projects that contribute directly to the bottom line of the business. Usually these are new products or new features.
2. Internal IT Projects - Internal projects have the goal of making the organization more efficient. Examples of these types of projects usually involve internal tools or automation.
3. Changes - DevOps defines changes as work. An example of making changes is deploying code to production or tweaking configuration.
4. Unplanned Work - This is the type of work organizations should always strive to minimize to zero if possible. Examples of unplanned work are production incidents and work being sent backwards in the value stream. Unplanned work is detrimental to the productivity of an organization.

A decorative graphic on the left side of the page consists of a vertical green bar. To its right, there are two columns of circles. The first column has circles in dark blue, light green, dark green, dark green, blue, blue, and blue from top to bottom. The second column has circles in light green, light green, dark green, blue, blue, and blue from top to bottom.

THE SECOND WAY: FEEDBACK LOOPS

In order to accelerate the flow of work in an organization, we need to introduce and amplify feedback loops throughout the value stream. The faster a developer or any member of the team receives feedback on broken builds, failing tests, or features that don't meet requirements; the faster the issue can be rectified and the work can continue to flow. We want to create feedback loops that inform from right to left in the value stream. You will often hear the term 'shift left', which means shifting the feedback as far left or as early in the value stream as possible. This is an extremely important concept because the earlier problems are discovered, the less expensive they are to fix.

To create some of these feedback loops organizations often rely on Continuous Integration pipelines that provide immediate feedback on problems with new code. This works by running tests and creating builds whenever developers check new code into the repository. Developers are then alerted of the status of their latest check in. As organizations become more mature they can move to Continuous Deployment, which is having a build ready to deploy at any time. Ultimately you can move to a Continuous Delivery model, in which code flows from a developers machine all the way to production without any manual intervention.

Feedback loops should not only exist in the workflow pipeline, but also in production environments. It is extremely important to have good metrics around how your products are behaving in production so that you can respond to your customers needs. From a software and operations perspective, we are usually concerned about the performance of our production applications. A good set of metrics to use for monitoring production workloads are the 4 Golden Signals. These consist of Error Rate, Latency, Traffic, and Saturation. Production monitoring allows you to obtain real time feedback on problems and respond to issues before your customer even knows they exist.



THE THIRD WAY: CONTINUOUS IMPROVEMENT

The natural state of all processes is entropy, and software development is no different. The Third Way calls for creating a culture of continuous experimentation, learning, and improvement within an organization. An organization should look to carve out time for employees to experiment outside of their daily duties, Google is famous for allowing employees 20% of their time for non business project work.

Another key tenant of The Third Way is shared responsibility. A healthy culture avoids pointing fingers or playing the blame game when things go wrong. In fact, a DevOps organization will look at failures as opportunities for improvement. Organizations like Netflix accelerate the pace of production failures using tools like chaos monkey, which purposely takes down infrastructure in an effort to force improvement in software resiliency. This allows them to survive entire AWS region outages.

The Third Way also subscribes to the idea that repetition is mastery. An organization must practice certain scenarios in order to get good at them. An example of this is production incidents. All organizations suffer production incidents, but how the organization responds can vary greatly. The Third Way advocates for a concept called Game Days in which production like outages are created and the team can practice responding in an orderly fashion. The goal of this effort is to reduce Mean Time To Resolution for outages, which is another key metric of highly functioning DevOps organizations.

CONCLUSION

In this article we touched on the symptoms of an organization in need of DevOps, the key principles of DevOps, some misconceptions, and DevOps origin. While DevOps is a great philosophy and can help improve the performance of your engineering department, it is a more holistic approach that requires the buy in of leadership at the highest levels. If you only apply DevOps philosophies to your engineering department, you may find that while you are improving engineering bottlenecks, you are ignoring other organizational problems that may be the primary bottleneck within your value stream. It is important to apply The First Way: Flow to your entire organization and adopt a product lead approach to software development.



**WE ARE ARTISTS.
WE ARE ENGINEERS.
WE ARE INNOVATORS.
WE ARE CALLIBRITY.**

Learn more about Callibrity
www.Callibrity.com

ABOUT CALLIBRITY

Callibrity is a software consultancy specializing in software engineering, digital transformation, cloud strategy, and data-driven insights. Our national reach serves clients on their digital journey to solve complex problems and create innovative solutions for ever-changing business models. Our technology experience covers a diverse set of industries with a focus on middle-market and enterprise companies.

Callibrity (kə'librətē) is a mashup of two different roots, calli and caliber. Calli means 'beautiful' in Greek, as in Calligraphy - beautiful writing. Caliber means 'a degree of merit or excellence'. We strive to do beautiful work with a high degree of merit and excellence.

 by **developers**
for **developers** 