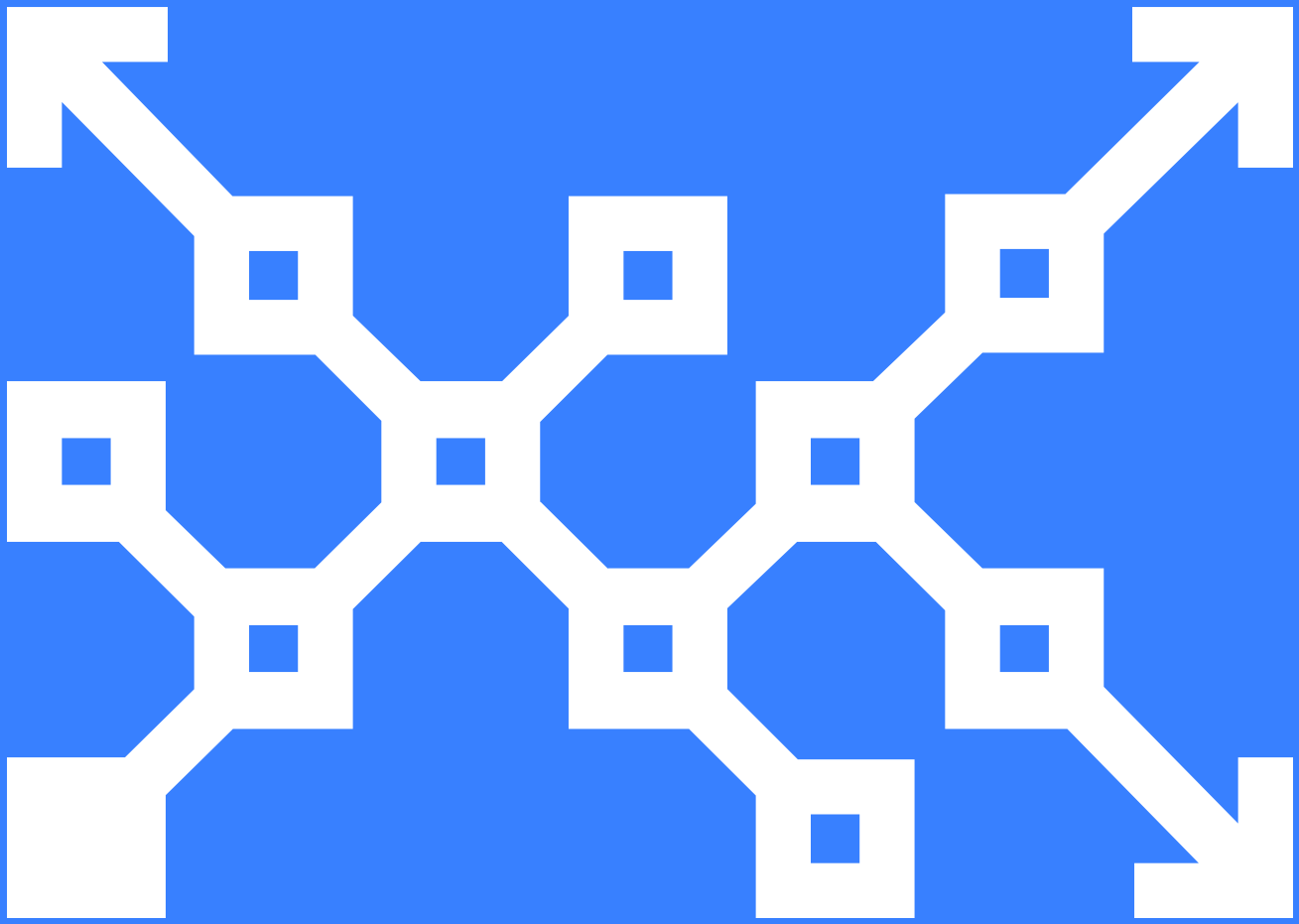
WHITE PAPER

Get Started with DevOps: A Guide for IT Managers



puppet

Contents

Intro

If you're managing an IT team, you've already heard about DevOps. It's a term that encompasses lots of things, and in many businesses, government organizations, nonprofits and educational institutions, DevOps is being discussed with some urgency as the path to faster delivery of software, staying competitive and becoming more efficient.

Our goal for this white paper is to help you tease out the whys and hows of DevOps as you work with your own team, with other teams in your organization (especially software development) and with upper management. We're pointing you to issues we've found are common in many organizations, and to things we've found can make a DevOps initiative successful.

Working with your operations team

As an IT manager working in operations, you hopefully spend most of your time with your operations team. If your organization is looking to adopt DevOps practices, you'll need to think about how to communicate to your team why things are changing, and how their usual working practices are likely to alter.

Why change is needed

There are two main drivers common to most organizations looking to adopt DevOps: a desire to move faster and a desire to scale up. Alongside this is often a desire to move away from a dysfunctional mode of working and towards a way of working that is more sustainable.

- **Moving faster.** Customers are ever more demanding, and technology has reduced the barrier to entry for competitors in many markets and industries. With the maturing of agile software development practices, development teams are now able to deliver new features faster. But departments using traditional tools and processes can find it difficult to keep up with the new pace. DevOps is about extending the advantages of agile into operations and the rest of the organization.
- **Scaling up.** The explosion of software as a service and infrastructure as a service, along with efforts to digitize many internal and external processes, means more and more computers and services are needed. With greater compute power, however, comes greater responsibility — these computers, devices and services need to be managed, maintained, secured, and monitored.

It goes without saying that a desire to move 10 times faster, and to operate at a 10-times-greater scale, rarely goes hand in hand with the option of increasing the operations team to 10 times its size. This means operations teams feel increasing pressure. Often that leads to a growing rift between operations teams, which are incentivized to keep things stable, and development teams or other business units that are incentivized to produce more, and at a faster pace.

These misaligned incentives, and the resulting rifts and barriers that develop between different teams, result in organizational silos and the growth of shadow IT: services that are utilized outside the management, and often the knowledge, of the operations team.

Table 1: Comparison of IT performance metrics between high[1] and low performers

The annual State of DevOps Report has consistently shown that IT performance metrics are much better in organizations that employ DevOps practices than in organizations that don't. This chart shows the differences between high-performing IT organizations and low performers; the critical difference between these groups (besides the metrics) is the extent to which they employ DevOps practices.

	2015 (Super High vs. Low)	2014 (High vs. Low)
Development Frequency	30X	30X
Development Lead Time	200X	200X
Mean Time to Recover (MTTR)	**168X**	48X
Change Success Rate	**60X**	3X

1 We used the same hierarchical clustering technique as we did last year, but to dig into the larger high-performing group that emerged this year, we did additional analyses and discovered a group of super-high performers.

Alignment with the organization's purpose

A core element of DevOps is alignment of each group's work with the strategy and mission of the overall organization. As an IT manager, a big part of your responsibility is making sure that your team understands how its work helps to move the organization forward. You also need to communicate to other managers around the company about how your team's work supports their teams and the overall mission. And you need to make sure that upper management understands the value your team contributes to the organization's mission, and how critical your team is to everyone's success.

What is DevOps?

DevOps isn't a fixed methodology or process; it's a community of practice and a set of principles. DevOps advocates Damon Edwards and John Willis described DevOps in 2010 with the acronym CAMS: culture, automation, measurement and sharing. Let's look at each of these elements.

Culture

First and foremost, DevOps is a cultural undertaking; it's about much more than embracing new tools and practices. It's about setting expectations and priorities, and the fundamental beliefs that guide these expectations and priorities. The 2015 State of DevOps Report shows there's a strong link between culture and successful management practices. We recommend you read the report's deep dive into the characteristics of a successful DevOps culture.

Automation

Automation is a given in any team that has embraced DevOps; it's just the way everything is done. Through automating common and repetitive processes, we aim both to make more time for higher level work and to open up opportunities for innovations that wouldn't be possible otherwise.

Measurement

Integrating feedback into work is a fundamental part of agile and lean practices. For mature DevOps-minded organizations, this means measuring absolutely everything that moves in production, and sharing those measurements with the widest possible audience. When embarking on a DevOps transformation, however, this can feel a long way off. So start with a smaller number of critical measures that map to business outcomes — for example, deployment frequency, lead time for changes, mean time to recovery (MTTR), change fail rates, etc. Metrics from the live service are useful for much more than just keeping the service running: They can show you and your team where there are opportunities for quick wins that will benefit the organization's mission.

Sharing

Today's complex organizations and software require teams of people with different skills and specialized knowledge. In order to work efficiently, however, it's important to work well together. Sharing facilitates this. One way to share more early on is by exposing metrics to everyone, as described above. There are also benefits to sharing outside the organization, whether that be contributing to open source projects, talking about your experiences at conferences, or writing public blog posts. Your team gets to share pride in its work, and you can also open dialogue with people outside whose ideas and experiences can benefit your organization. Sharing tools, skills, information and your successes are all part of building an effective organization.

When identifying and making changes to how you and your team work, it's useful to think about those changes in terms of culture, automation, measurement and sharing. Base your choices on maximizing opportunities in these areas, and you should find yourself well on your way to breaking down silos and improving your organization.

What change might look like

Here we'd like to describe some things you might do differently — things that other organizations have done on their path to success with DevOps. You might find you're doing some of them already, which is great. Remember, DevOps isn't all new; it's really a collection of things that have worked for others faced with challenges similar to yours.

Focus on a team over individual specialists

Any reasonably complex environment needs a wide range of skills to operate, and those skills can be distributed across a number of people. In some traditional teams, this leads to the development of silos, even within the team itself. For example, you can have one person looking after the database, another in charge of network hardware, one who's the Windows admin, another who's the Linux admin, and so on.

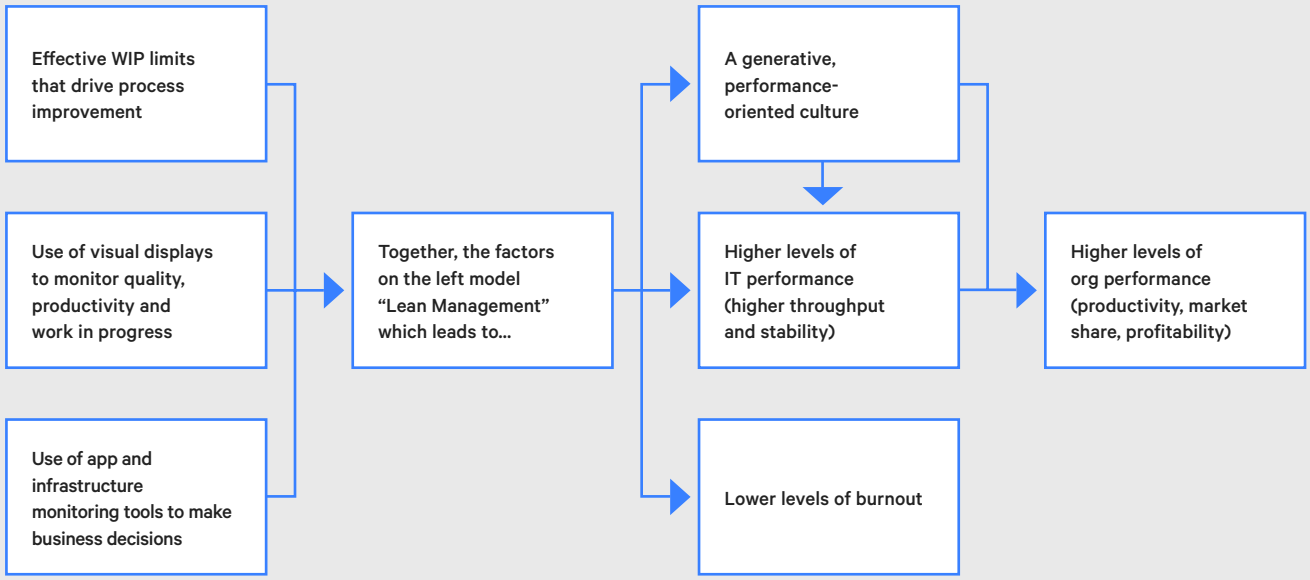
This level of specialization often means that to get anything done, several people have to be available at the same time and must synchronize their work. The overhead of this communication alone often takes longer than the work itself. This isn't to say individuals can't be responsible for individual services at any given time, but it's advisable to rotate those roles frequently so people can learn and widen their skills. It's important to develop a "fix it" attitude in the team, rather than a "report it" mentality. Prioritizing team learning and advancement over individual advancement tends to help combat the development of silos, leading to greater efficiency — and higher morale, too.

The concept of a cross-functional team is a common one in successful DevOps implementations, and it's an approach that has proved successful across many different organizations and industries, from insurance to healthcare to software development. A cross-functional team combines people from different disciplines — for example, putting software developers, system operators, quality assurance specialists and product managers on the same product team. You can see the alignment with DevOps right away: It's impossible for Dev to toss code across the wall to Ops and walk away when they're all working on the same team every day.

In a cross-functional team, it is not individuals who own the work, but the team itself. Because responsibility for outcomes is shared, the majority of people on the team tend to become more T-shaped over time: They have deep knowledge and expertise in a specific area, and at the same time, broadly understand the goals of the organization and the functional areas that support it. When the team is composed of such T-shaped people, there's greater empathy, because they understand more about each other's areas. The teams are also better aligned to the organization's strategic goals, as these have been emphasized over individual achievement and team-only goals.

Specialists, rather than owning their own area, often act as domain experts and help to advise and educate. This approach works to avoids single points of failure (everyone likes to go on vacation sometimes) as well as upskilling everyone on the team. That doesn't mean the team doesn't have specialists or that certain tasks don't require specific skills, just that you're looking for T-shaped people. This often has the knock-on effect of lessening the burden of on-call rotations, as people have wider domain expertise and can more easily share work.

Figure 1: Path diagram showing relationships between lean management practices, IT performance, culture, burnout, and organizational performance.



Infrastructure as code

Operations teams that successfully embrace DevOps invariably take the approach of describing much of their infrastructure, and their interactions with it, in code. Successfully embracing infrastructure as code means you will be replacing static and only occasionally updated documentation with living source code (for example, using Puppet's domain specific language). Now you can also implement other common software development practices for managing infrastructure, such as code review, continuous integration and unit testing. These practices can make your infrastructure far more reliable while opening it to others on the team, so they can understand and help manage systems, too.

For operations people, the infrastructure-as-code approach means embracing new tools and new ways of working. For team members who have some development background, this can be quite familiar. But for a more traditional systems administrator, this can be a new experience, perhaps even a little intimidating at first. However, it's not about every sysadmin becoming an experienced programmer who can write complex code; it's about adopting, piece by piece, some of the software processes and practices documented elsewhere in this report.

Embrace agile ways of working

Once you're focusing on a team rather than a collection of individual specialists, you need a way of planning work that allows for visibility into what is being done and for planning ahead. Especially with operations work, which can be quite interrupt-driven, it's important to factor in incident management or support issues.

Story: Systems engineers promote DevOps at Disney

When the Web Operations group at The Walt Disney Co. changed its name to Systems Engineering, the renaming made the team see themselves and their roles differently — and it changed their colleagues' view of the team, too. "The software engineers now saw us as kindred engineers," said Jason Cox, director of systems engineering. "We got more collaboration, and more sense of what the software engineers did. We saw more of their space and their pains, and what it takes to write apps. We were able to tell them about resiliency, and what it takes to run apps.

Get more of the story:

Disney's DevOps Journey: A DevOps Enterprise Summit Reprise

While adopting DevOps practices, it's common to find developers embedded in and working with the operations team.

Depending on the size of the team, it's sometimes desirable to use different methods for different kinds of work. You need one method for longer-term build-out work — the type of work that requires focus over weeks or months, and normally has a deadline associated with it. You need another method for more interrupt-driven work, maintenance work and other smaller tasks. This might mean adopting something like scrum for the planned work, and kanban for the rest. Done well, this allows for limiting work in progress (so work can get done, not just started), as well as providing an idea of velocity or flow that you can use to judge capacity and work effectively towards any deadlines.

A few examples of activities often found in agile processes include:

- **Limiting work in progress.** A common, but somewhat counterintuitive practice in some agile methods is to actively limit the number of different tasks being worked on at any given time. The intent here is to move individual tasks through to completion more quickly, and to encourage collaboration in order to improve quality and overall productivity.
- **Working in iterations.** Many agile techniques employ a fixed-length iteration, often "sprints" of about two weeks. The idea is, rather than trying to plan out in minute details all possible tasks for the next six months, to plan the details that are needed for the next few iterations. This helps you improve the process and can lead to a better product, because the learning from any one iteration can be used to plan future iterations. The technique of shorter sprints works best when coupled with bigger-picture planning over a longer period of time.
- **Regular retrospectives.** A retrospective is time you set aside to ask the whole team what is working and what can be improved. The idea is to see the process you're using as something you can iterate on, in addition to iterating on the work you're doing. The best approach to that process improvement is to make many small improvements, rather than doing a big-bang reorg. The Agile Retrospective Resource Wiki has lots of different ideas for session formats.

Cross-team working

While adopting DevOps practices, it's common to find developers embedded in and working with the operations team, and for systems administrators and other operators to be embedded for a time in development teams. This cross-pollination should be encouraged wherever possible. Having developers dedicated to the operations team for a period of time will help the ops team embrace some aspects of automation. Developers often have experience with some agile ways of working, such as pairing or unit testing, and can share these practices with operations people. Embedding operators on development teams helps devs to think more about how their applications will run in production, and encourages them to improve logging, monitoring, deployment practices and more.

DevOps is about much more than just developers and operators, though. Encouraging cross-pollination with other specialists in your organization will also improve processes and results. Pairing up with someone from a customer-facing part of the organization, for instance, can help technical employees better understand the business reasons for changes and pressures.

The concept of a cross-functional team is a common one in successful DevOps approaches. Here the team, rather than individuals, own the work.

New roles and skills

Most people will tell you DevOps isn't a job title or a new role on the team. But that's not to say that new roles or new skills won't be required to fully embrace some of the advantages that DevOps brings. You'll increasingly see job titles like "site reliability engineer," "web operations engineer" or "infrastructure engineer." In many cases, these are simply the evolution of systems administrator roles rather than anything materially new, but in some cases, changing titles can have a positive impact on the perception of the team. When it comes to acquiring new skills, though, there's no doubt that programming skills are key to adopting infrastructure as code, and DevOps in general. It's not about everyone on the team becoming a software developer, but it probably does mean everyone on the team will need to acquire more software development skills — for example, learning to employ automated tests, using source control, and making use of continuous integration. It's worth noting that you may have all or many of these skills in your organization already. A manager's role is to find the people with those skills, and foster them. Here are a few things to look for:

- Teams or people who have already successfully automated specific services.
- Products or projects in your organization that are working particularly well.
- Developers who are particularly interested in infrastructure and DevOps.

Managing organizational change

As any good manager knows, understanding which practices you want to adopt is just one part of implementing organizational change. Managing that change successfully is at least as important as the changes themselves. In this sense, adopting DevOps practices is no different from any other change, and any good literature on organizational change management should be worthwhile.

The following are common concerns amongst operators, and specific to adopting DevOps practices in operations teams:

- **Fear of automation.** Adopting DevOps practices usually means embracing automation as a default solution to many problems. This raises the concern of being "automated out of a job." In reality, because the intention is generally to go faster at greater scale, hardly anyone is ever automated out of a job. It's more often about automating tasks away, eliminating wasted work, and increasing the amount of time each person spends on high-value work.
- **Increased risk when moving quickly.** The focus of many operations teams has traditionally been stability, and operators tend to be good at assessing risk. Going faster sounds like it should increase the risk, leading possibly to an unstable infrastructure, loss of reputation and even the loss of financial bonuses. You can help allay your team's fears by explaining how various DevOps practices actually minimize risk, and by providing examples that are relevant to your organization. Yes, you could point to Facebook or Google or other so-called unicorn companies, but in fact, even companies in traditional industries such as banking, manufacturing or insurance are achieving 30 times more frequent deployments, 200 times faster lead time for changes, 60 times fewer failures, and 168 times faster recovery times, according to the 2015 State of DevOps Report. It's not really about the industry or organization you're in; it's about implementing practices that deliver better results.

Because the intention is generally to go faster at greater scale, being "automated out of a job" is very rarely the case. It's more often about automating tasks away, and increasing the amount of time each person spends on high-value work.

Where to get started

Getting started with any transformation is the hardest part. Three things are essential for any DevOps initiative:

Get buy-in from the team

DevOps began as a movement of practitioners, and in many cases, successful adoption of DevOps practices originates with the developers and operations people themselves. You can help your team adjust to the ideas behind DevOps by finding out which parts appeal most to people. Make space to discuss these ideas as a group. Look for quick wins to build momentum and win people's confidence in the new practices. Giving your team some autonomy, space and time is much preferable to trying to execute on a management dictate.

While getting buy-in from the team is certainly necessary, this year's DevOps Report showed us it's not sufficient for success. Management has to support DevOps, too, and the manager who actually works with the ops and/or dev team plays a critical role. This is the person who can connect the organization's strategic goals to the work his or her team does every day, and help the operators or devs align their work to the business strategy. These managers can also communicate the value of their teams' work to higher-level management, and make sure their people have the space and time they need to learn the new ways (and to fail).

A desire to go 10 times faster and operate at a 10-times-greater scale rarely goes hand in hand with the option of increasing the operations team to 10 times its size.

Increase visibility

Having visibility into what everyone on the team is doing — how work is progressing and what blocks work from getting done — is critical to introducing other DevOps practices. In fact, the 2015 State of DevOps Report discloses that IT performance can be predicted in large part by the extent to which teams create and maintain visual displays that show key quality and productivity metrics, plus the status of work in progress. When all this data is available to everyone, the team can take ownership more clearly, and individuals are less likely to feel swamped — another benefit of visualizing metrics and work in progress. A simple Kanban board on a wall near the team, or in a software tool, is a good start.

You'll probably be surprised at first by just how many things the team is working on. This visibility can help lead to greater control over work in progress. This in turn leads to the team being able to react better to changing circumstances, and to a more sustainable pace of work.

Reach out to other teams

DevOps is about collaboration, and that has to start somewhere. Building trust with your fellow managers is one of the most effective ways to remove roadblocks for your team. Here are a few ways to start:

- Ask if members of the operations team could each shadow a developer for a few days, listen in during daily standups, or attend sprint retrospectives.
- Invite developers to postmortems for outages, and to your team's planning sessions. While the dev team manager is certainly welcome to attend, it's more important for the devs themselves to hear about the fundamental issues your team deals with in the normal course of their work.
- Publish your team's progress reports publicly to the entire organization, and encourage everyone to read them. Asynchronous collaboration and awareness can have a huge impact on improving processes.

Working with the development team

For an IT manager working in operations, one of the most important aspects of DevOps is achieving a much closer working relationship with the development team, both at a management level and at the level of the ops team's day-to-day work. This section covers how you can get there and what it can look like.

Why change is needed

Everyone has war stories about the development team completing the build of the new application or website and heading off to celebrate, leaving the operations team to work out how to get it into production at the last minute. When the new thing finally goes live, it turns out to be slow and customers complain, but never mind, it's an operations problem now. This is unfortunately an all-too-common story, but the immediate reaction — to blame the cowboy developers — isn't fair, either. They likely achieved what they were tasked with (most likely hitting a deadline), but the scope of that effort and the incentives supporting it just weren't aligned with those of the operations team.

The larger problem here — bigger than the problematic deployment of a new application — is the negative feedback cycle, and the resulting lack of empathy, between development and operations. This lack of a collaborative relationship generally leads to contract-style negotiation around SLAs, and barriers being erected between teams which then further distance and damage the relationship. There really is another way.

The problem is the negative feedback cycle, and the resulting lack of empathy, between development and operations.

What change might look like

No one size fits all

The first thing to realize is that there are many different ways of succeeding, partly because there are many different types of organization. Maybe the operations team is much larger than the development team, or vice versa. Maybe operations is partly outsourced, or all development is done by a third-party firm or firms. Maybe the development team is based on a different site from the operations team, or even in a different time zone. These differences definitely mean one size doesn't fit all, but some of the underlying principles are applicable to whatever model you have.

Collaboration over contract negotiation

One of the tenets of the agile manifesto is "customer collaboration over contract negotiation." This should definitely be applied to the relationship between development and operations teams. Rather than your team spending its time building barriers that need to be jumped (or gamed), use that time to work with the development team on the actual problem. For instance, rather than holding a series of meetings to set some minimum performance measures, pair with the development team to help them measure, and then improve, the application's performance. Pairing is a great low-barrier-to-entry way of starting collaboration across teams. And the dev team will likely be happy for the help.

Security is a great icebreaker

The nature of operations tends to give ops people a healthy fear of real-world users, and a good eye for security issues. Many operations teams will have security specialists, or at least people with some experience of conducting an investigation or combating an attack. Security is an area that many developers have a growing interest in, but little experience with. So make security an olive branch. Reach out to the development team to offer expertise. Conduct internal training sessions, and invite developers into threat briefings. Show them monitoring data around malware or DOS attacks. The important part here is giving people common information on which to build relationships, and learning to appreciate each other's expertise.

Building self-service platforms

Rather than operations playing the role of gatekeeper, it's increasingly common for the operations or infrastructure team to be an internal service provider. Rather than all deployments being done by the operations team, ops can provide a standard way for the development team to deploy their own applications. Rather than the operations team being the only ones with access to production monitoring, they can maintain the monitoring system and provide self-service access and training to development teams.

Remember, this transformation won't happen overnight; it will require patience on the side of the skilled operators to help review configurations and to help upskill the teams using the service. There is also the question of service level agreements, and managing monitoring and changes of the managed service itself.

This shift to some self-service fundamentally changes the dev team's relationship with the operations team from ops having a gating function to ops empowering devs. Start by identifying largely repetitive work that is ripe for standardization, and then design a service model around that activity. Repeat for other common activities.

You built it, you run it

Another commonly held belief in the DevOps community is that the best people to support a complex application in production are the people who actually built it. This means costly handovers between teams can instead become opportunities for training. The real impact of this change is to align incentives between the people building the application and those running it. Because the development team knows they will be directly responsible for any problems in production — for example, by taking turns being on call — they are much more likely to prioritize work to make the software operable, stable and resilient to failure. So work on monitoring instrumentation, on ease of deployment, on logging, etc., are no longer an afterthought, but a necessary and normal part of application development.

Rather than operations playing the role of gatekeeper, it's increasingly common for the operations or infrastructure team to be an internal service provider.

This doesn't mean that many of the traditional operations skills are no longer needed — quite the opposite. It might mean the team structures change, and that the organization moves towards cross-functional teams composed of developers and operators working together. The organization may also decide to have a smaller central operations function that provides shared services, an internal consultancy, and standards.

Table 2: How to create a generative culture

Studies of organizational culture have found that workplaces with a generative culture — as opposed to a pathological or bureaucratic culture — are productive and better for people working within them. The characteristics of a generative culture map nicely to DevOps practices.

Characteristics of a Generative Culture	DevOps Practices
High cooperation	**Cross-functional teams.** Many organizations create cross-functional teams that include representatives from each functional area of the software delivery (business analysts, developers, quality engineers, ops, security, etc.) This allows everyone to share the responsibility for building, deploying and maintaining a product.
Messengers trained	**Blameless postmortems.** By removing blame, you remove fear, and by removing fear, you enable teams to more effectively surface problems and solve them. Mistakes happen. Holding blameless postmortems is a valuable way to learn from mistakes.
Risks are shared	**Shared responsibilities.** Quality, availability, reliability and security are everyone's job. One way to improve the quality of your services is to ensure that devs share responsibility for maintaining their code in production. The improvement in collaboration that comes from sharing responsibility inherently reduces risk: With more eyes on the software delivery process, it's a given that some errors in process or planning will be avoided. Automation also reduces risk, and with the right tool choice, can enable collaboration.
Bridging encouraged	**Breaking down silos.** In addition to creating cross-functional teams, techniques for breaking down silos can include co-locating ops with the dev team; including ops in planning throughout the software delivery lifecycle; and implementing ChatOps2.
Failure leads to inquiry	**Blameless postmortems.** Our response to failure shapes the culture of an organization. The more you focus on the conditions in which failures happen, as opposed to blaming individuals for failures, the closer you'll get to creating a generative culture.
Novelty implemented	**Experimentation time.** Giving employees freedom to explore new ideas can lead to great outcomes. Some companies give engineers time each week for experimentation. Others host internal hack days or mini-confrences to share ideas and collaborate. This is how many new features and products have originated, and it shows how much value employees can generate for an organization when they're released from habitual pathways and repetitive tasks.

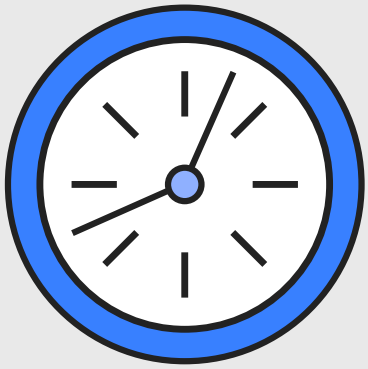
Managing change

Managing change within the development team is likely to center around balancing ability with responsibility. Simply handing over the keys to production in any reasonably complex environment isn't likely to end well.

- **Emphasize non-technical operations work.** Developers regularly underestimate or overlook the non-technical side of operations and service management. So it's important to help with things like capacity planning, incident management or auditing requirements. This can probably involve internal training, or even better, pairing with experienced members of the operations team. Pairing is better because many people learn more easily that way, and the close personal interaction helps to develop empathy between colleagues and teams.
- **Involve the development team in the design of shared services.** When moving to a model of providing shared services, you must involve the end users. Simply taking existing processes and tools and declaring they are now shared services won't work. Start with a clear idea about user needs, and from there work out what's possible in the short and medium term. What the operations team want out of a deployment mechanism might not align exactly with the needs of the development team, so having everyone involved should lead not just to a better service, but also to a better understanding of the underlying problem and the tradeoffs chosen in the process of solving it.
- **Include developers in on-call rotations.** This can be tricky, and can take some time to get right. But the benefits over time can outweigh the initial risks. The reasons for caution tend to be mainly logistical — Is this covered in existing contracts? Are there time zone differences between existing teams? Are there existing SLAs? — or personal (being on call is no one's idea of fun). So it's important to talk about why this is happening, and to remind developers it's their software, and even more so once it's running in production. The main benefit here is an alignment of incentives: it makes conversations about shipping software with known performance or stability problems harder, which is much better than the alternative — shipping substandard software to production and then discovering its operational problems.

Where to get started

- **Start with developers who show an interest.** Although over time most, if not all, of the development team should have some level of experience with operations, that will take time. The aim isn't to convert developers into systems administrators, it's to teach skills, and to engender empathy for operations as a specialty. It's much easier to get started with people who show an interest in learning aspects of operations. Look for team members of all seniority who show an interest in DevOps, infrastructure as code and automation, and start with them. Over time, those individuals will be able to help others. Depending on the developer's interests, monitoring dashboards, configuration management or automated testing are good places to start.
- **Monitoring, deployment and continuous integration.** Monitoring, deployment and continuous integration are all good places to start with shared services that will be of utility and interest to the development teams. Depending on the enthusiasm within the organization, it may be best to pick the one that will bring the biggest benefit, or the one where you have the most mature solution in place already.



Any business case for a DevOps-influenced transformation will often ask for time at least as much as for money.

Not just development and operations teams

Don't get hung up on the word DevOps. The movement is much broader that just those roles, and depending on the size and shape of your organization, you may have other silos that closely relate to the service delivery process. Much of the above applies equally to quality assurance teams, network operations, security specialists and support teams. Start by identifying interactions between teams that are often painful or time consuming, and work to automate or provide a service offering to make these interactions easier in the future.

Working with management

Any change program requires buy-in from management, and DevOps is no different. But why should your direct manager and others in management back this particular effort?

Whether you work in a large organization with a formal business case process, or a smaller organization where you still need to convince management, it's important to present DevOps as a business change, not a technology-driven change. While DevOps is often associated with new tools, the real change comes from a new way of working. Any business case for a DevOps-influenced transformation will often ask for time at least as much as for money. As well as showing bottom-up enthusiasm for the proposed changes, it's important to point to success in other organizations and to trusted external opinions. Depending on your organization, the trusted opinions could come from industry analysts or the technical press. You might point to success stories from similar organizations, or to industry surveys like the DevOps Report.

Don't use the DevOps word

DevOps is a buzzword. That's not a bad thing; in fact it's part of the reason for the movement's success. Having a single easily identifiable term has made it possible for a community to coalesce around various practices and a desire for better collaboration between different business functions. But when talking to management, it is generally much better to talk about solutions to problems, and about practical benefits. It's important that managers and teams align on why they're embarking on this journey in the first place. Talk about:

- Improving instrumentation and monitoring, which leads to identifying problems before customers report them, improving mean time to recovery.
- Deploying software in smaller batches and more frequently, reducing risk of failure and getting new features to customers quicker.
- Managing configuration in code, allowing changes to be rolled out quickly and accurately, and with less manual work.

These are all valuable practices in their own right, with beneficial outcomes, and so are likely easier to talk about in a business case or budget meeting than trying to explain the higher-level concepts of DevOps.

Collaboration and responsibility

One challenge when explaining some of the tenets of DevOps to management is managers' fear that no one will be held responsible. It's important to shift day-to-day responsibility to a team, rather than to individual practitioners, to adopt practices like blameless postmortems and to provide self-service internal services where possible. But a lot of that can be viewed as passing the buck. "Collaboration is all well and good, but who do I blame if this doesn't work?"

The simplest and easiest answer is, as the IT manager pushing the change, to take the responsibility on yourself. So for any change, the person to turn to is you. This can feel scary, and obviously comes with a degree of risk. But it also makes space for your team to embrace DevOps practices, and hopefully the material presented here will help convince you that success is not just possible, but likely. And with risk comes the rewards of success; you get to own these, too.

DevOps is a buzzword. That's not a bad thing, in fact, it's part of the reason for the movement's success.

Getting time for the team to create new processes

By its nature, DevOps can't really be constrained to just one team, as that team likely interacts with other teams in a multitude of ways. This means new processes must evolve for those interactions, and part of selling DevOps to management is buying time to implement these new processes. For example, you might need to institute a requirement for development colleagues to take part in postmortem meetings after outages (hard to plan ahead for, and important to hold as close to the incident as possible). Or it might be a move towards a self-service model of operations that pushes some responsibility for provisioning and deployment out from the central operations team.

The best advice here is to not change everything at once, and wherever possible to talk early on with the managers of the other teams affected. You don't want to schedule a glut of meetings, but you do want to give people an opportunity to discuss and provide input. This also has the advantage of involving those other teams in the design of the service, which should make it more effective.

Conclusions

When looking to introduce DevOps practices into your organization, remember the acronym CAMS: culture, automation, measurement and sharing. DevOps is fundamentally about changing culture, about building much more responsible organizations that can move quickly in ever-changing circumstances. It's also about taking full advantage of modern technology, to replace manual, time-consuming and error-prone activities with automation. But in automating existing processes, you'll likely open up entire new opportunities for working smarter, too. One of the reasons for the success and interest in DevOps is that a central tenet is to measure the impact of everything, and using those measurements to make better decisions in the future. It's often important to stress the metrics-based nature of many of the common DevOps practices; it will be easier for people outside your team to get behind something if you have hard evidence that it's working. And finally, DevOps is an opportunity to share. To share knowledge, to share skills, to share experiences — all with the aim of improving not just the output of your work, but also the way you work.

DevOps is an opportunity. Despite all the buzz, the conversation is just starting, and although organizations of all sizes are demonstrating the ideas and approaches that work, it's still very far from being the way everyone works today. In other words, this is the perfect time for an organization to leap ahead by adopting DevOps.

Further reading

DevOps is a topic of great interest at the moment and new content is being published all the time. There's also lots of good relevant material that predates the DevOps banner. Here's a short list of useful reading for anyone who wants to go further.

- **Lean Enterprise.** Stories from large enterprise organizations adopting agile and lean approaches to delivery.
- **Continuous Delivery.** An in-depth guide to continuous delivery, explaining what it is, why it's important and techniques for implementing it in your organization.
- **Visible Ops.** A solid introduction to service management. Good fundamentals, especially for people coming from a development background.
- **The Phoenix Project.** A novel that explains DevOps and other practices by telling the story of a fictional, but all too believable, organization struggling with outdated approaches to IT.
- **Web Operations.** An excellent collection of essays on subjects from monitoring and metrics to blameless postmortems and collaboration between teams.

About the author

Gareth Rushgrove is a senior software engineer at Puppet. He works remotely from Cambridge, U.K., building interesting tools for people to better manage infrastructure. Previously he worked for the U.K. Government Digital Service, focusing on infrastructure, operations and information security. When not working he can be found writing the DevOps Weekly newsletter or hacking on software in new-fangled programming languages.