puppet

The 5 Stages of DevOps Evolution:

A Guide for CIOs

Contents

For the past seven years, we've examined the relationships between IT performance, DevOps practices, culture, organizational performance and other factors that affect organizations' business outcomes. Our research has consistently broken new ground in understanding how DevOps leads to business success.

We've amply demonstrated that IT is a key value driver in an era where speed, agility, security and stability are all critical to succeeding — and to remaining competitive. Today's CIO can't accept tradeoffs. It's perilous to sacrifice stability and security in favor of faster releases, but if you sacrifice speed and agility to security and 99.9999 percent uptime, your competitor could get to market first.

This year we designed our DevOps survey to reveal what successful organizations actually do as they progress on their DevOps journeys. We discovered that they experience five distinct stages of DevOps evolution, which we have shared in full detail in our **2018 State of DevOps Report.**

This CIO guide digests our research into a report that's just for you. We want to help you quickly understand what successful organizations do at each stage of DevOps evolution; which practices must be established at each stage (we call these the defining practices); and how you can best support your team throughout its DevOps journey to assure continuing progress and success.

Before we describe each stage, here are some principles to keep in mind:

- **The journey to DevOps isn't linear.** Every organization will have its own distinct path to success, based on starting point, industry and business model.
- **Listen to your teams.** Our research showed that team managers and practitioners have a stronger grasp than CIOs of how much DevOps progress has been made. Listening to the people who work with day-to-day technical issues will show you how best to enable their work as they introduce and improve DevOps practices.
- **Start with the practices that are closest to production.** One of the biggest pain points for most teams is deployments. Early success with improving deployments improves morale and establishes foundational practices that help teams quickly advance to higher levels of DevOps evolution
- **Automating security policy configurations is critical to reaching the highest levels of DevOps evolution.** Your team's momentum will stall unless security is involved in technology planning and design, and building security into your technology requires the automation of most, if not all, security approvals.
- **Your endorsement and promotion of sharing and collaboration are incredibly important.** DevOps can't take root and grow without a culture of sharing and collaboration. You should encourage your teams, right from the beginning, to share and showcase their successes, because this will help to grow DevOps thinking and practices in your organization. You can also help spread the word of these successes and of other teams adopting the practices, to show just how important sharing and collaboration are to your organization's success.

Throughout this paper, we refer to development teams and operations teams, because that's the usual way people write and talk about DevOps. However, in most of the scenarios we discuss here, you could just as easily substitute the term "application team" for "development team."

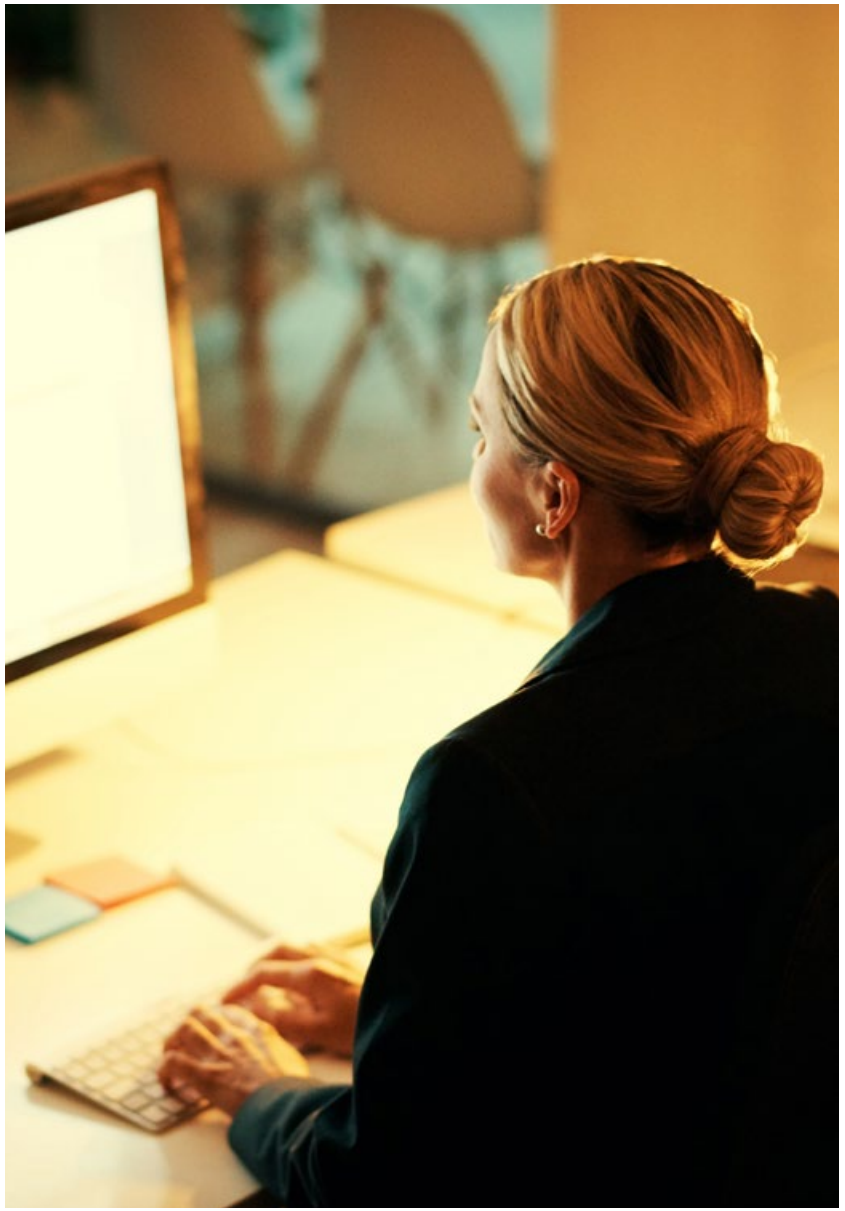
The stages of DevOps evolution

Stage 0: Build the foundation

Successful teams start their DevOps journey with the knowledge that increased collaboration and sharing will help solve most problems with IT systems and software delivery — a principle at the very heart of DevOps.

Our research shows that organizations achieving the highest levels of DevOps evolution work from the beginning to establish five foundational practices, all of which promote collaboration and sharing. Of all the DevOps practices, these five have the most significant impact across the entire DevOps evolution:

- The team operating a service can configure the monitoring and alerting for that service.
- Deployment patterns for building applications or services are reused.
- Testing patterns for building applications or services are reused.
- Teams can contribute improvements to tooling provided by other teams.
- Configurations are managed by a configuration management tool.



The most highly evolved organizations are the most likely to engage in these foundational practices, not just at the beginning but all the way through their DevOps journey. For example, highly evolved organizations are *24 times* more likely than the least-evolved organizations to enable teams to define their own monitoring and alerting criteria for applications and services in production.

Your teams should start small with each of these practices, and evolve them as they adopt other DevOps practices. We provide some interesting examples of how it can work, and a deeper discussion of these practices, in **The 5 Foundational DevOps Practices: How to Establish and Build on Them**.

How you can help

This foundation-building stage is critical to the evolution of DevOps. Your support is extremely important as your teams adopt and perfect new practices that dissolve barriers between teams via collaboration and sharing.

You can best support this stage if you:

- Listen with an open mind to the problems your managers and their teams are trying to solve, whether these are technical in nature or involve communication, measurement and sharing.
- Foster and facilitate measurement, automation, collaboration and sharing — both tools and processes.
- Encourage teams to start small, and celebrate the achievement of these small initial goals.
  - One good example is time synchronization, which allows distributed systems to communicate reliably and securely. This may not sound like much, but it can have a huge impact on efficiency and driving down error rates.
- Ask how teams will measure the outcomes of automation and increased sharing and collaboration, and make sure you are included in all metrics reporting and progress updates.
  - Get proactive: Establish a small task force to evaluate and standardize monitoring and alerting practices. This will demonstrate just how important this practice is to you, and to the cultural shift you are promoting.
- Understand that every attempt will not succeed, and publicize the importance of learning from failure.
  - Demonstrate the importance of learning by mandating blameless retrospectives after incidents, and by encouraging sharing of outcomes from the retrospective.
  - Use your position to create space for agile practices to thrive and grow, and make sure the managers and team leaders in your department understand this is part of their job, too.
  - Pay attention to the pockets of success in your organization and nurture them.

- Reward the behaviors that will promote DevOps evolution.
  - Use your regular communication channel to team members to call out examples of teams reusing and sharing deployment and test patterns. Don't have a regular communication channel yet? Create one.
  - Point to significant milestones as the team rolls out a configuration management system.

"There are two approaches to changing the culture — top-down and bottom-up. Top-down involves leadership promoting automation as a strategy. Bottom-up is encouraging people to think about automation as an opportunity rather than a threat."

Sai Billanuka, Vice President, Technology THOMSON-REUTERS

Stage 1: Normalize the technology stack

Most teams have a lot of complexity in their systems at the outset of their DevOps journey. That complexity creates chaos. So this stage is reactive; it's about comprehending the extent of complexity and creating a plan for reducing it. This stage prepares teams for the next, proactive stage: standardizing technology.

For successful teams, two practices define this stage of DevOps evolution:

- Application development teams use version control.
- Teams deploy on a standard set of operating systems.

Version control is vital to further progress for both development and operations teams. Its adoption is the first step towards continuous integration and continuous delivery, which are required for faster, more reliable delivery of high-quality software.

It's important to note that agile development practices pressure operations teams to deploy changes faster. As the CIO, you need to be sensitive to this, and push for the changes in IT operations that will help alleviate that pressure.

Normalizing the technology stack is one such change. Eliminating redundant systems and standardizing on a smaller set of operating systems clears the way for the ops team to adopt consistent automation and management patterns, saving time on patching, tuning, upgrading and troubleshooting various systems. Another benefit: your security teams will be happy because limiting the number of OS versions also reduces the number of vulnerabilities they have to track.

Along with these two defining practices, successful teams also:

- Build on a standard set of technology.
- Put application configurations in version control.
- Test infrastructure changes before deploying to production, helping to eliminate errors and increasing efficiency.
- Make source code available to other teams, enabling teams to contribute back to the code while helping to establish sharing as a core cultural value.

How you can help

At this stage, you should:

- Endorse broader adoption of version control, both for development and operations teams.
- Ask your team leaders to review and consolidate operating system versions by a specified date, with the initial goal of eliminating outliers.
  - Team leaders should describe why outliers exist and provide a replacement plan. The teams managing outlier systems probably want to get rid of them, but haven't had the time or space to do so.
  - This is a great opportunity for operations to start exercising the muscle of working with development and business teams to replace or eliminate existing systems.
  - Recognize that some consolidation efforts will be easier than others. For example, you may be running RHEL and SLES because different teams prefer different versions of Linux, but you could easily standardize on one version. Sunsetting Solaris, on the other hand, may require rearchitecting your application, which requires more time and effort.
  - Ask your teams to codify how you adopt new technologies, including operating systems. They should be able to answer: "Can we use a technology we're already using today? Dogma aside, does the benefit of adding a redundant technology to the stack outweigh the cost of learning, managing and scaling it? What will we eliminate to add this new technology?"
  - Make this project a priority, make it visible and be sure to celebrate the team's work.
- Have your teams decide which technologies (besides operating systems) should become the standard for your organization, with an eye to easy cross-team and cross-organization sharing, as well as future needs.

Your consistent endorsement of sharing across technical teams is extremely important. De-siloing responsibilities and knowledge is a core DevOps value, and your enthusiasm for breaking down boundaries will help establish sharing as a fundamental cultural value for your organization.

"As we moved to a more Agile development methodology, we've been able to get a much higher degree of productivity out of our developers by having everybody using microservices, standardize on the same framework, use the same set of tools. We've gone from a traditional waterfall release cycle of four times a year with one major release every 12 months to a cadence that is much more in line with Agile, where we are doing anywhere from 10 to 15 releases a month. That's something we'd never be able to do without having a common set of tools and common set of standards across the corporation."

- Barry Libenson, CIO, Experian

Stage 2: Standardize and reduce variability

Organizations at this stage take a proactive approach to the chaos caused by complexity. Both development and operations teams continue to standardize the technology stack. The technology choices made at this stage will affect the organization's future success.

Successful teams at this stage have two defining practices:

- Deploying on a single standard operating system.
- Building on a standard set of technology. This includes items such as databases, key value stores, message queues, identity stores, etc.

At this stage, teams also:

- Reuse deployment patterns for building applications and services.
- Rearchitect applications based on business needs.
- Extend version control beyond application code and configurations to include system configurations.

Collaboration continues to grow at this stage. For example, a dev team can collaborate with ops colleagues to select a single database or identity-store technology, so operational considerations are taken into account.

The benefits of this stage are both immediate and long-term. New applications and services can be deployed faster with reused patterns, and with significant reduction in errors. Shared patterns established at this stage can be improved and evolved over time, setting your teams up for success as they expand DevOps practices to other teams — and ultimately, the entire organization.

How you can help

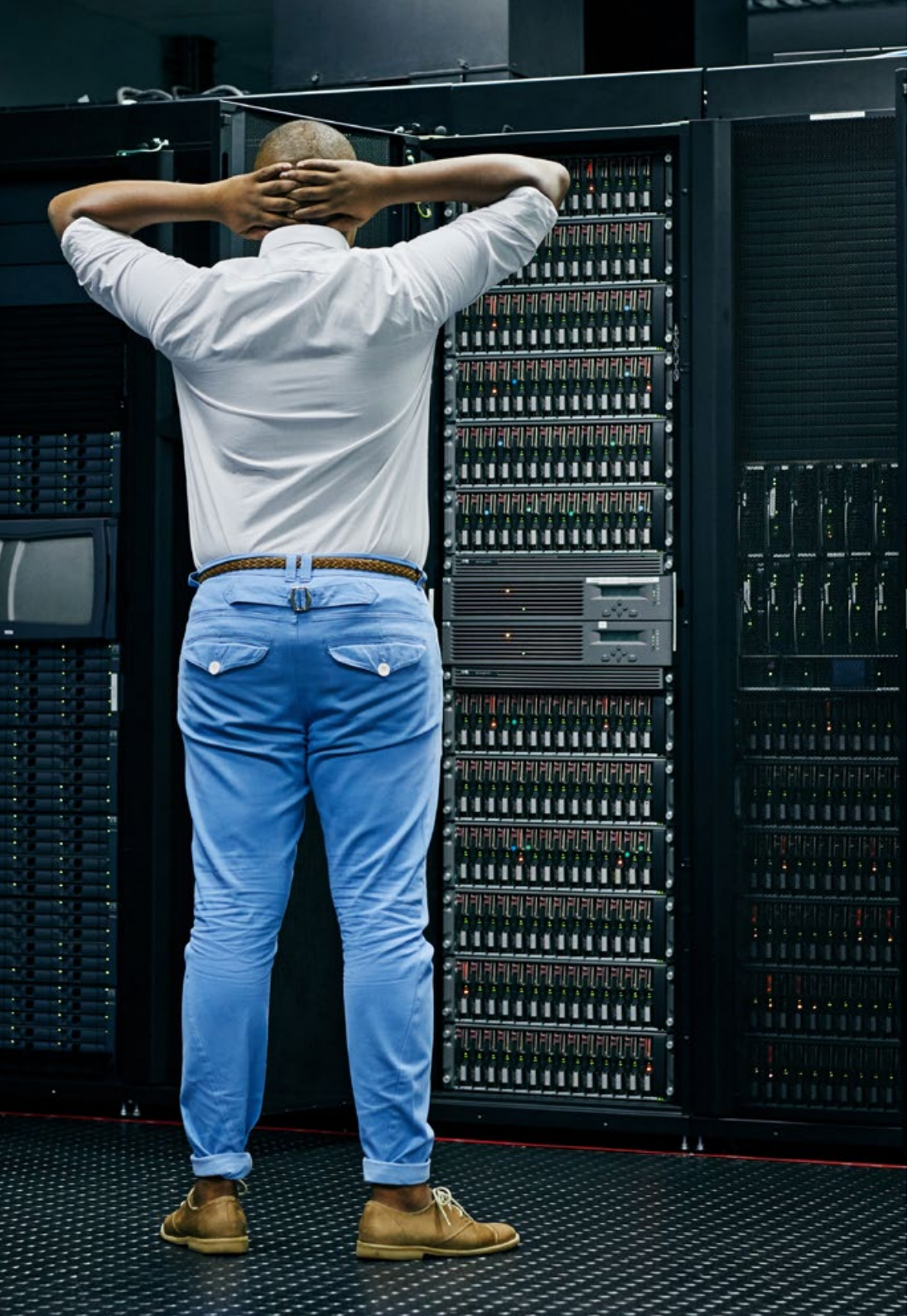
To rationalize your organization's current toolset, tell your development and operations team leaders to catalog and prioritize capabilities — for example, operating-system configuration, software configuration, deployment, patch management and IT compliance reporting. Next, map the tools the teams are currently using for each capability to determine where there's overlap, so you can eliminate redundant tools.

We know one CTO who requires his teams to list only the tools they actually like. This filters out 80 percent of all tools and makes the ultimate choice of a single tool for each purpose (or set of purposes) much easier. This CTO says that once the choice has been made, it's much faster to roll it out, because there are already internal champions for the tool.

Michael Guggemos, CIO of IT solutions company Insight, took another approach, following what he calls the "law of large bodies." For example, "If 70 percent of our environment is a Cisco network, our standard is now Cisco," Guggemos said.

This is a time to actively push for greater cross-team collaboration, and to track it. Ask for reports on outcomes of collaboration, and for metrics on deployment cycle improvements.

As always, it's important to applaud achievements. At this stage you should call out the results of increased collaboration as much as you call out improved metrics for deployment cycles and software quality. Because rationalizing tools is so important, you could also announce when a tool has been selected, talk about why, and congratulate the teams on their selection process.



Stage 3: Expand DevOps practices

Now that teams are collaborating around technologies and deployments, it's time to address a problem that comes up frequently at this stage: The development team's throughput outpaces the delivery team's ability to deploy. The danger of not addressing this discrepancy quickly is that business-focused outcomes won't improve, and people will feel their efforts didn't pay off as promised.

Two practices define this stage:

- Individuals can do work without manual approval from outside the team.
- Deployment patterns for building applications and services are reused.

These practices speed deployment while improving predictability and reliability. That in turn builds trust in the new methods for both your dev and ops teams, not to mention business leaders in your organization (including you).

The reuse of deployment patterns at this stage provides building blocks for the infrastructure automation that happens at the next stage.

Additional practices and milestones that lead to success at this stage:

- Individuals can make changes without significant wait times, accelerating time to value.
- Teams continue to build on (and streamline) a standard set of technologies.
- Post-incident reviews are held and results are shared.
- Continuous integration is in regular use.
- Infrastructure teams are using version control.
- Service changes can be made during business hours.
- Infrastructure changes are tested before deploying to production.

How you can help

To make progress at this stage and beyond, individual team members must be able to work without manual approvals from outside the team. You should remove outmoded bureaucratic processes by telling your change approval board (CAB) to reconsider every approval they require.

Start by asking the CAB to work with the teams they represent to determine which changes can be removed from the list fairly easily, and then lay out the steps needed to eliminate these approvals. Essentially, you want to put the CAB out of business.

The reason CABs frustrate people so much, and make it so hard to accomplish anything quickly, is that they often review and approve things like DNS changes, clock updates and other items that teams can review and approve for themselves. Yet you may not want to entirely eliminate outside change reviews. So if you do choose to retain the CAB, it should review and approve only high-impact, major changes, and particularly those that will occur at large scale — for example, replacing core network switches in a primary data center.

That said, we'd like to emphasize that plenty of successful organizations manage changes successfully without a CAB.

Hiring, retention and DevOps

Our 2014 DevOps research showed that IT performance is lower in organizations that use CABs, and higher in organizations where technical teams rely on peer review to verify code quality. We also found the existence of a CAB rarely results in a lower change failure rate, or shorter service restoration times. So relying on a CAB reduces throughput without actually improving quality or uptime.

Outside of change approval processes, ask all your team leaders to identify the top three reasons for wait times. Then prioritize working on the reasons that are mentioned most frequently.

To sum it up, removing bureaucratic process will:

- Promote more efficient workflows.
- Build capacity amongst team members.
- Create favorable conditions for further improvements.
- Improve IT performance.
- Increase loyalty among employees who want their expertise to be trusted and valued. (And that's most employees.)

In most markets, it's not easy to hire highly skilled technical employees, so attracting and retaining these workers is a top priority for most CIOs and CTOs.

Our 2016 DevOps research showed that the highest-performing IT organizations had the most loyal employees, as measured by whether they'd recommend their team and their organization as a great place to work. Other studies have shown that this willingness to recommend is correlated with better business performance.

DevOps practices themselves are highly correlated both with high IT performance and better business results. DevOps practices make IT teams more efficient and successful, helping people keep a reasonable work-life balance while restoring pleasure and learning to the work itself. This greater job satisfaction matters to the bottom line: Our 2014 research showed that job satisfaction is the top predictor of organizational performance. DevOps practices also help avert burnout, a common reason why technical people leave jobs.

Post-incident reviews are important, so mandate them, as well as sharing of the results. Following each review, you should publicly applaud the lessons and any improvements that result from reviews (these improvements may come later, of course). This will signal to your teams how important reviews are, and that their true purpose is learning and improvement. For more information on post-incident reviews, we recommend this **post-incident review template**.

You might feel you should attend post-incident reviews. Don't. Your presence will change the tone of the event, and people are likely to feel uncomfortable about being completely frank in the presence of the big boss (that's you).

Instead, make sure you get a report on each post-incident review and the actions that will be taken. You can ask attendees for any clarification you need after you read these notes. Make sure you respond to these post-incident review reports with encouragement. You can publicize to the wider team, or even the entire company, the benefits that come out of specific reviews (especially when the incident was experienced by others outside the technical teams). It's a way of spreading the value of learning from unexpected events.

As it becomes possible to make service changes during business hours without disrupting customers, the business or employees' work, you should publicize this widely within the organization — not just to the technical teams.

"With approximately 5,500 properties in 110 countries, global automation is not a nice-to-have, it's a must-have for us. We open an average of two hotels a day in some part of the world, so establishing global standards from the ground up is critical. We establish global standards for bringing in technology and IT services and implementing best practice standards for operations. We automate as much as possible with the right tools and processes. By standardizing development and processes, we are able to provide a consistent level of service that is repeatable for all brands."

Sudhir Menon, vice president of enterprise information management, Hilton Worldwide

Stage 4: Automate infrastructure delivery

Here's where successful teams solve the problem of developer throughput outpacing operations' ability to deploy — an outcome that many consider to be one of most important gains from a DevOps initiative.

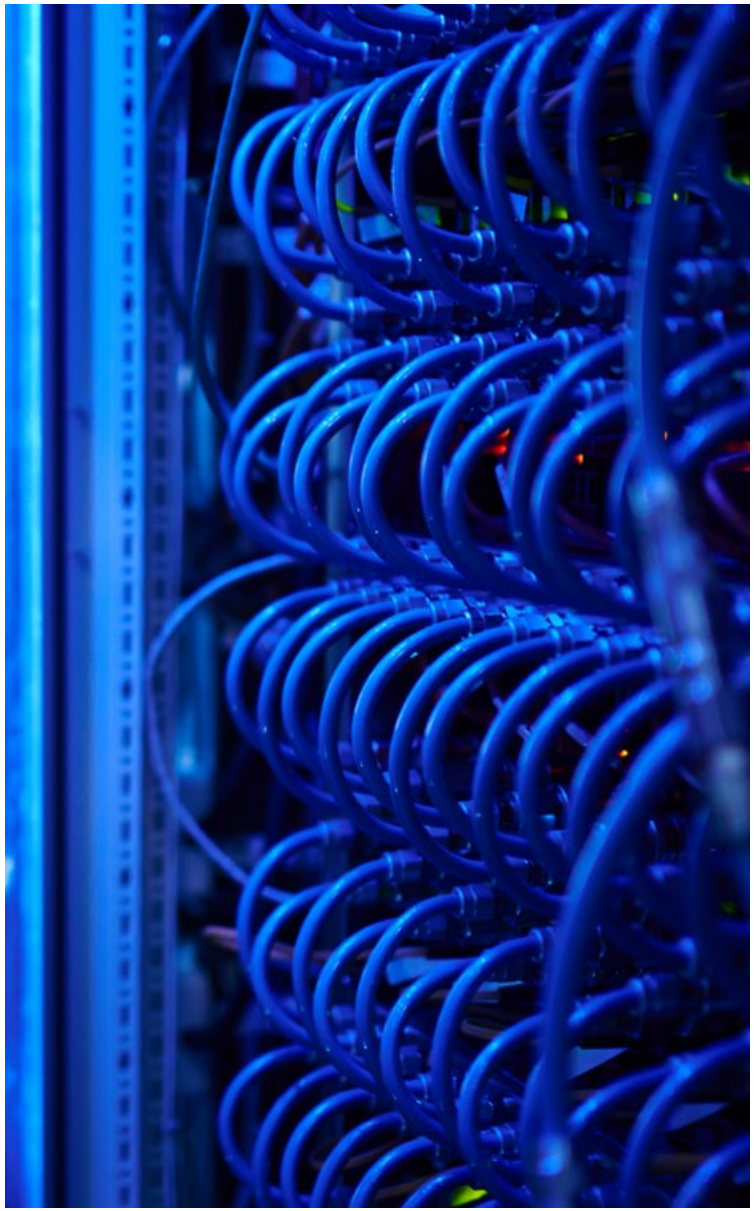
Two practices define this stage of DevOps evolution:

- System configurations are automated.
- Provisioning is automated.

Automating systems configurations and provisioning not only allows ops to deploy code faster, it also allows both developers and QA teams to work in environments that match production, leading to fewer production errors. Achieving this level of automation also builds the foundation for creating self-service across the organization, which broadly improves efficiency, accelerates time to value and delivers greater satisfaction to more workers.

At this stage, successful teams have accomplished several important milestones:

- Both system and application configurations are in version control.
- Security policy configurations are automated.
- At least some resources are available via self-service.



How you can help

The focus of automation work at this stage needs to be consistency, not just speed. Infrastructure automation not only provides consistency in the immediate, it also provides the foundation for creating reusable deployment and testing patterns, and for continuing to build on these.

The operations team must have your support for taking the time to automate configuration and provisioning. So refrain from initiating other projects right now that would require Operations' time and attention. In fact, the team will need to cut back its work with other issues and improvements to get this automation project done. They may also need additional hands for regular operations work.

If you don't intend to hire more people to facilitate automation, then as the CIO, you must make it known you've given permission to let some things go unresolved at this time.

However you decide to carve out time for this automation work, it will be worth it: moving to a known platform for automation will result in higher levels of service.

Encourage your teams to first build self-service systems for their own use, before building self-service for other teams. This will let them learn from any failures in a safe environment, where the customer is more tolerant of technical failures.

If you've been tracking your deployment frequency, lead time for changes, and mean time to recover (MTTR) regularly, you should now see a marked improvement. If you don't, it's time to talk to your teams about why that's the case, in the same open and blameless manner used for incident reviews. (If you haven't been tracking these metrics, now is a good time to start.)

If security policy is now automated, you should also be able to point to fewer incidents or scares, and celebrate the greater harmony and efficiency that have resulted from this change. As always, take time to publicize and explain the value of your team's accomplishments, especially for other teams — for example, how any new self-service saves time for specific teams outside of your space.

Stage 5: Provide self-service capabilities

Teams can work much faster and more efficiently now, as app developers can deploy testing environments on their own, and provisioning is automated. Success metrics for projects are widely visible, and experiences and lessons are shared both internally and externally.

Two milestones define this stage of DevOps evolution:

- Incident responses are automated.
- Resources are available via self-service.

At this stage, known good processes have made it possible to expand self-service to more teams — not just technical teams, but business teams as well.

The advantages for these other teams are time savings and greater efficiency, plus the satisfaction that results from being able to do your job without waiting for someone else. For the operations team, self-service means that consistency is built in — services are delivered as they should be, with risk mitigation built in, too. This makes life easier for Ops, too.

Security is more fully integrated into the development process at this stage:

- Security policy configurations are automated.
- Security teams are directly involved with technology design and deployment.

Successful teams at this stage are also engaged in rearchitecting applications based on business needs.

How you can help

Security and automation

The involvement of security early in the software development cycle should be a high priority at this stage. We discovered that C-suite respondents to this year's DevOps survey thought, to a significant degree, that security was more involved with technology design and development than team leaders and practitioners said it was. So make sure your security team is actively participating, and ask for project reports that will show you whether this is the case (or attend project meetings, sitting quietly and listening).

The items to prioritize for security are:

- Automating incident responses.
- Automating security policy configurations.

These will provide a very high return for your organization, saving employee time and, for some organizations, increasing revenue flow.

Don't be complacent. Our DevOps research this year showed that C-level executives have a much more optimistic view of DevOps progress than team leaders or practitioners. For example, 54 percent of C-suite respondents said security policies in their organizations were automated, compared to 38 percent of respondents at the team level. Even more — 57 percent — reported that incident responses were automated, compared to 29 percent at the team level. So make sure you continue to ask penetrating questions, demand progress reports, and respond to them with both questions and appropriate praise.

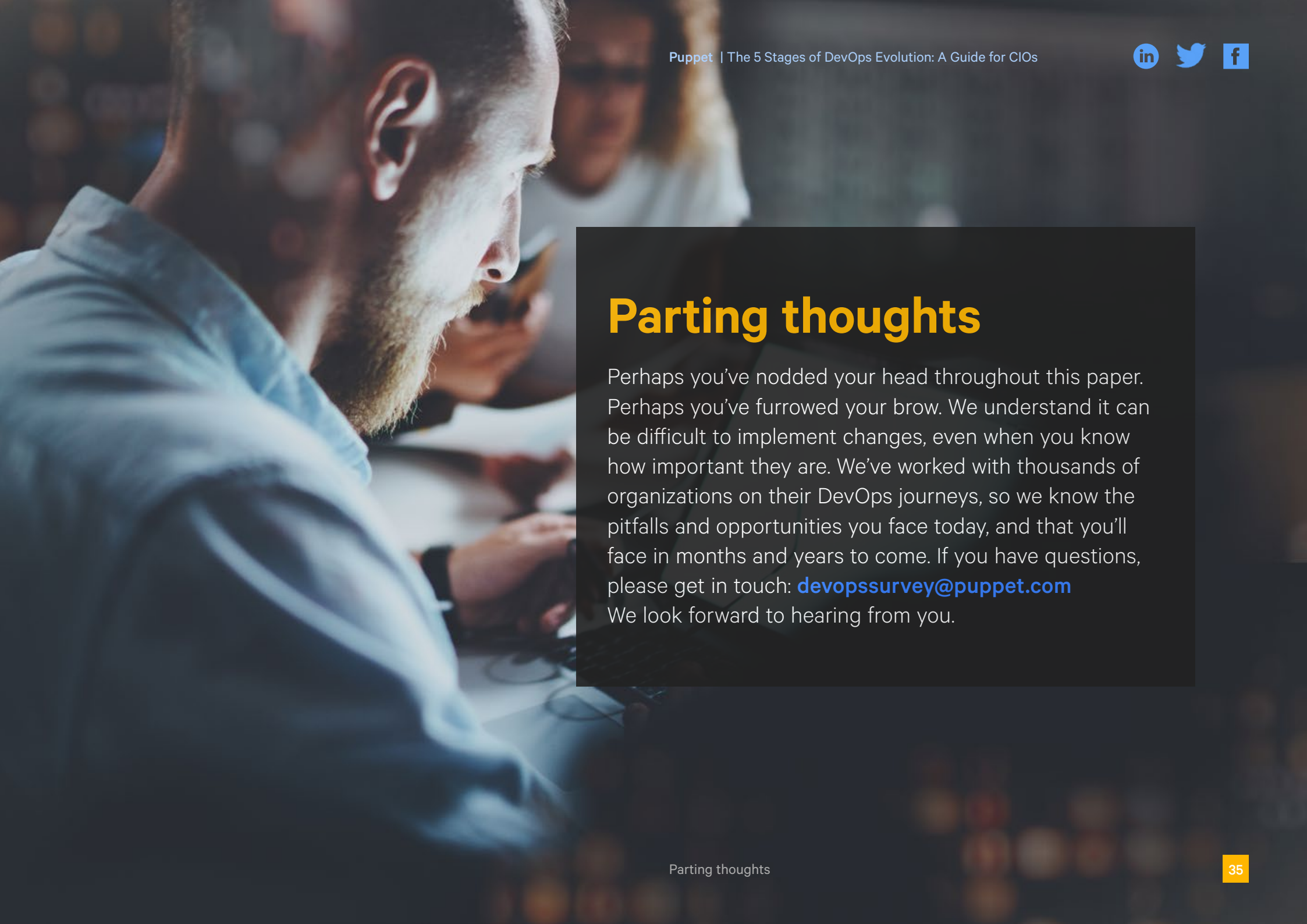
Self-service

After your teams have created self-service for their own use, encourage them to offer self-service to adjacent teams. (For example, the server team should be able to handle its own network and storage provisioning.) Next, provide self-service to business teams that need application instances, test instances, and more.

Make sure your teams understand the importance of standardizing self-service (just as your organization has standardized deployment and testing patterns), and set that expectation from the start. The concept is to create a single user experience. Think about how self-service so often grows organically out of immediate needs: to get service from multiple teams, there are multiple interfaces and processes with little to no knowledge of each other. You don't want this to happen.

Publicly thank the teams that have provided self-service capabilities to other departments across the organization. Even better, share data showing how much faster it is now to get a service than it was before your DevOps initiative kicked off. This will help build awareness, enthusiasm and support for your team's contributions to the entire organization. And it will make it much easier for you to get the resources your team needs for further innovations.

You should point out how improvements in applications and services better serve the business itself, by cutting out waste or responding to customers faster or more accurately. Sharing these improvements raises morale for everyone in the organization, by showing that they work for a company that can successfully innovate and compete. Everyone likes knowing they're part of a winning team.



Parting thoughts

Perhaps you've nodded your head throughout this paper. Perhaps you've furrowed your brow. We understand it can be difficult to implement changes, even when you know how important they are. We've worked with thousands of organizations on their DevOps journeys, so we know the pitfalls and opportunities you face today, and that you'll face in months and years to come. If you have questions, please get in touch: **devopssurvey@puppet.com**
We look forward to hearing from you.