

Getting Started with DevOps



puppet

The shortest path
to better software.



Contents

Introduction	3
Step 1: Discovery	9
Step 2: Standardization	20
Step 3: Automation	25
Step 4: Continuous improvement	29
Step 5: Driving innovation and enabling the business	32
Step 6: Measurement	39
Finding time: Let it burn	47
Resources	50

A person is seen from behind, sitting at a desk in an office environment. The desk has a laptop, a smartphone, and a mug. The entire scene is overlaid with a semi-transparent yellow filter. The text 'We live in a technology-driven world.' is centered over the image.

**We live in a
technology-driven
world.**

Software is on our wrists, on our walls and in our cars. It's changed the way we shop, the way we work and the way we stay connected.

In this world, every organization is a software company, and the demand for useful applications is relentless.

User expectations are higher than ever, and applications that work well today are out of date tomorrow.





Traditional development methods just can't keep up. Backlogs balloon, projects are postponed, and outside consultants are brought in to meet deadlines. To break free of this cycle, you need a better way to work.

Though DevOps is not a cure-all, it's a critical part of the answer, bringing development and IT operations together to create a streamlined system for software development and deployment. More a method than a prescription, DevOps is a collection of engineering, behavioral, and organizational practices focused on moving rapidly, safely and sustainably from idea to reality. To implement DevOps, any organization needs to make big changes — changes that impact culture, processes and people.



Culture

DevOps thrives in a workplace that cultivates trust across high-performing, cross-functional teams aligned around common business goals.

Process

DevOps replaces manual, siloed processes with automated ones that promote collaboration.

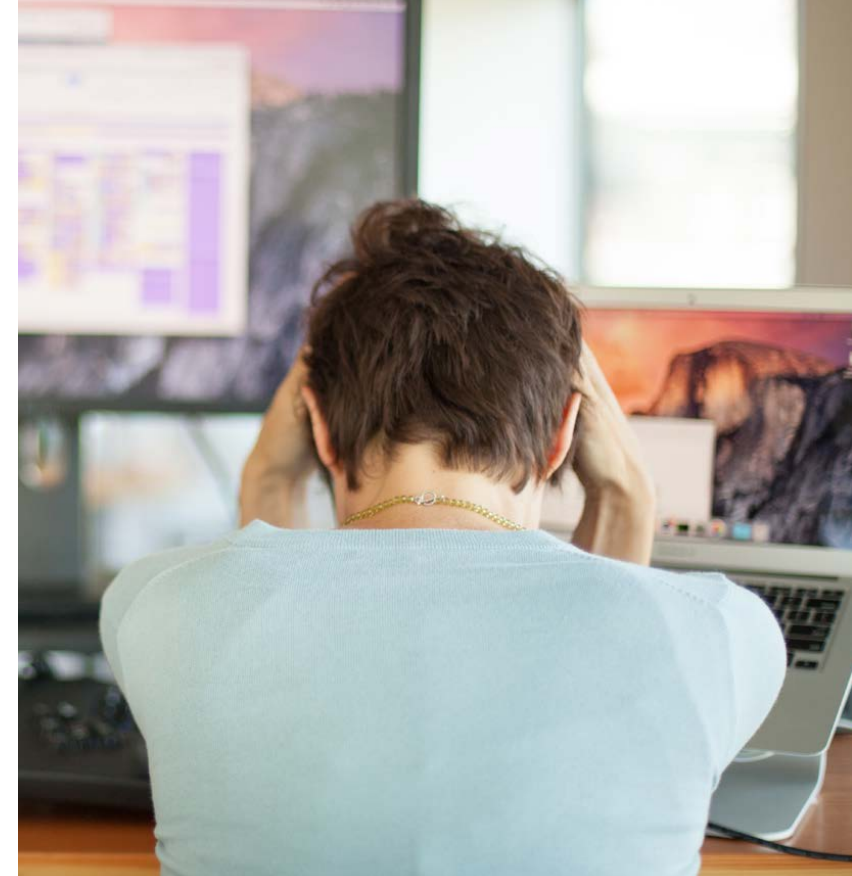
People

DevOps encourages people to grow and evolve their talents. Everyone in your organization works together to build an effective team.

Look, we get it.

Big goals like these can seem impossible to achieve.

Collaborating across teams and functions can be difficult, and it always seems easier to just keep doing what you're doing today. But as customer demands increase, applications become more complex, and fewer resources are available, the cost of ignoring these needed changes is high. More failures, slower recovery from those failures, and unhappy IT employees are what you can expect if you stick to the status quo.



“We have had some great results from our DevOps initiatives so far. We reduced our cost per release on one application by 97 percent, and by automating our testing, we’ve reduced multiple man-days of effort down to an overnight, hands-free process.”

Jonathan Fletcher,
enterprise architect and technology, platform & DevOps lead at Hiscox

The payoff with DevOps is tremendous.

You'll be able to respond to market changes and customer needs quickly and efficiently. You'll be able to run experiments and test new ideas without spending huge amounts of time and money. You'll learn faster, and become more agile and efficient. Your team will love their jobs again — and you will, too.

- DevOps is about delivering reliable products that run well in production, so you can delight your customers while delivering value to your business.
- DevOps practices help you deliver software faster at a higher level of quality, and with lower labor costs.
- DevOps optimizes processes by removing wasteful practices, promoting faster feedback and giving your organization more opportunities to learn and improve.
- DevOps makes it possible to measure IT functions that weren't measurable before.

So how do you build a fast, functional DevOps team?

The following step-by-step instructions will guide you through the process of bringing your teams together.

You'll learn about the tools and processes you need to build a strong DevOps culture in your organization.



Step 1: Discovery

In this stage you're on a fact-finding mission to understand the state of the tools and processes in your organization:

- What's working, and what isn't?
- What should improve to best serve your needs?

Start working team by team, process by process, and you'll soon find processes ripe for improvement — opportunities to create real wins and boost confidence throughout your organization.



Sam Eaton, director of engineering operations at Yelp, says you need to observe and participate before making any changes. “Get to know the people who are doing the practices as they exist now,” Sam advises. “Spend some time with them ... You should take the on-call rotation, take the alerts, and walk a mile in the other person’s shoes, before you speak up about what needs to be changed.”

This is a common practice at Yelp. Managers are often expected to work temporarily as individual contributors to get hands-on experience. Once a person has done that, “you can ask the tough questions,” Sam said. You’ve earned that right, because everyone knows you’ve actually done the work.

Once you start asking questions, you might find there are valid reasons for a current process. Perhaps even more important, as people explain why certain processes are in place, and what conditions were when the team started doing things that way, **“people will probably point out the problems themselves,”** Sam said. You’ll get a more accurate picture, and you’re also making it easier for people to acknowledge the need for change, “because they aren’t defending the choices they made back then.”

Because change makes people uncomfortable, and carries actual risk to operations, “you should be concentrating on evolution, not revolution,” Sam said.

Make sure to present change as both experimental and incremental. “Here’s something we can try, and if it improves things in a certain direction, then we can continue to iterate,” is a much less threatening approach than declaring a change will be permanent — regardless of how it turns out.



“You should be concentrating on evolution, not revolution.”

Sam Eaton
Director of engineering operations at Yelp



We recommend a specific set of questions for each team that will help you pinpoint where the best wins might be.

Asking these questions opens communication and gives people the reassurance that they're being heard. This listening is important work, especially in the early stages, and is vital to shifting your workplace culture.

Keep in mind that transformation can be difficult and slow, and typically requires perseverance,

both on the part of managers and technical team members. Don't set out on your discovery process expecting to fix lots of things right away. You need a long-range plan, and it won't all get done in a year — or even two. Accept that the long-range plan may also need to change as market conditions and the business change.

You'll be promoting a culture of continuous learning and the notion of change fitness, even in the discovery phase.

Once you have a culture of continuous learning and the flexibility to change, you'll be able to adapt to a changing marketplace faster — and better — than the competition.



Here's a tip!

First, work with your own team to **understand current processes and where improvements could be made.** Next, share what you learn with another team, and **encourage them to share potential improvements.**

Use these example questions as inspiration to create questions that are relevant to your organization.



Questions to ask your IT team

- How do you define a deployment?
- How painful are deployments? What's causing the most pain?
- How does a deployment work? Who is involved, and what do they do?
- What do your servers do, and what services are running on them?
- What's the most manual, time-consuming thing you do?
- Based on these answers, what should we tackle first?

Questions to ask your development team

- How long does it take to get the infrastructure you need?
- How long does it take for you to get a working environment?
- How soon could you replace your development environment if something happened?
- What are you using for version control? What is under version control?
- To what extent have you implemented continuous integration?
- Do you work in small batches, merging code into the mainline daily?
- What is the average size of a change set? Could it be smaller?
- How often do you build? Continually? Nightly? Less frequently?
- How long do builds take?
- Do you test your own code?

Don't forget about about quality and security.

You'll also want to talk to your quality assurance, test and security teams during the discovery phase. These teams need to be an integral part of the software delivery cycle; test-driven development (discussed in the [Standardization](#) section) is a key DevOps practice.

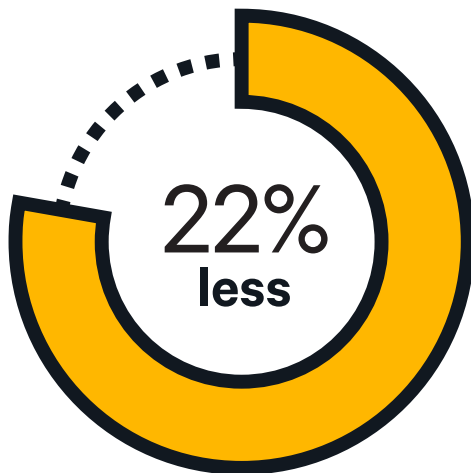
Many organizations mistakenly wait until the end of the software development cycle to incorporate security and testing. That's a way to end up with low-quality code, tons of security fixes and endless delays. In fact, it's common for teams to spend much more time fixing issues and performing security remediations than they did writing the code to begin with.

DevOps practice calls for including test engineers and security specialists in your planning process. Testing has to be automated, both for efficiency and to prevent costly errors. Test and security colleagues can help you write automated tests for compliance with your organization's own security policies and with industry standards (like PCI compliance).





**time spent
remediating security issues**



**time spent
on unplanned work and rework**

Teams that work security and testing into the early stages of their development cycles spend half the time remediating security issues that their peers do, and 22 percent less time on unplanned work and rework, as revealed in the [2016 State of DevOps Report](#).

They've moved both quality and security to the left — a concept borrowed from lean manufacturing, and familiar to anyone who's read the story of how Toyota was able to release better-quality cars to world markets long before U.S. auto manufacturers.



Use these example questions as inspiration to create questions that are relevant to your organization.



Questions for quality/test teams

- Are you included in the early stages of software planning?
- What tests are being done manually, and what is the role of manual testing?
- Which tests can be automated? How would we do that?

Questions to ask your security team

- Are you included in software planning meetings?
- Which security and compliance policies and processes are involved with application deployments?
- Which security and compliance policies and processes are involved with infrastructure deployments?

Talk to management, too

Conversations about DevOps are migrating from lunchrooms to boardrooms, as business leaders recognize that DevOps can help them pull ahead of competitors ... or that their competitors are already doing it. The business leaders in your organization can be powerful allies as you introduce and accelerate DevOps practices.

In fact, many tech teams find that management buy-in is critical to shifting organizational culture to become supportive of DevOps. As we've said, instilling DevOps practices takes time. If your management team understands how you're going to get there, and what kinds of early results they can expect, you're more likely to get the time you need to make the fundamental changes in both processes and human behavior that DevOps requires.

Asking the right questions will open a dialog with your management team, and help them understand that DevOps can solve some of their biggest problems.



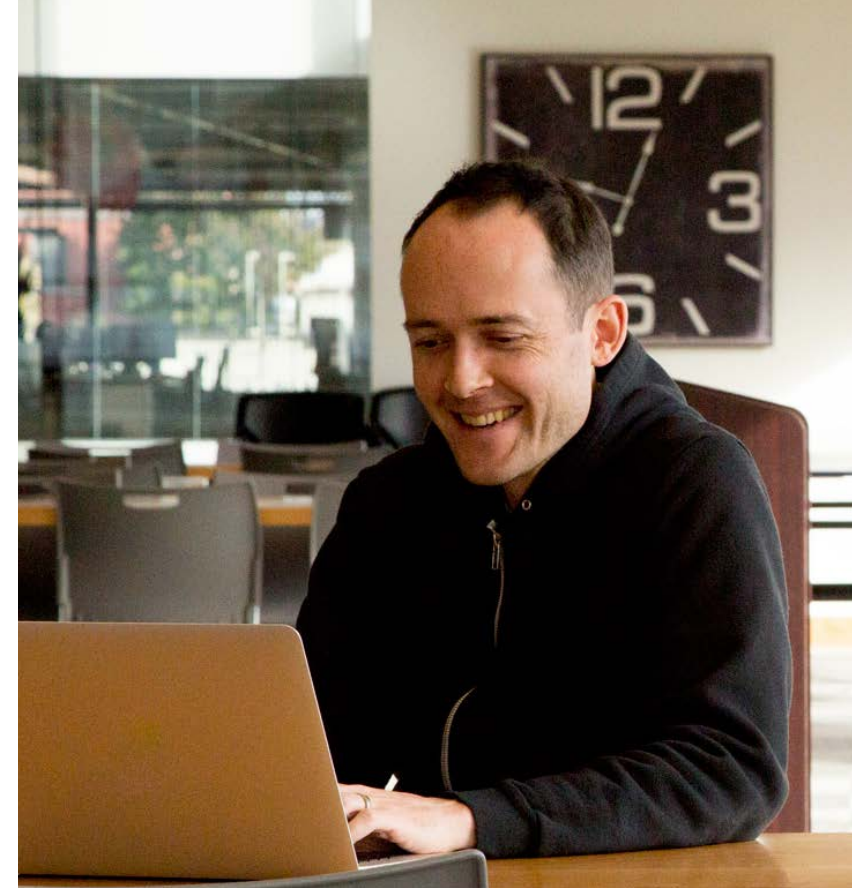
Use these example questions as inspiration to create questions that are relevant to your organization.



Questions to ask management

- How well does the cadence and reliability of software deployment support your business plan?
- How much visibility of software quality do you have across your division or company?
- How much visibility of IT infrastructure reliability do you have across your division or company?
- How quickly can your organization respond to shifts in the market?
- How competitive are we, compared to others in our market?

Once you have gained a clear understanding of the pain points and bottlenecks within your organization, you can choose which processes to automate first. You'll want to automate to free up people's time for other tasks, and to deliver early wins that will encourage technical employees and managers.



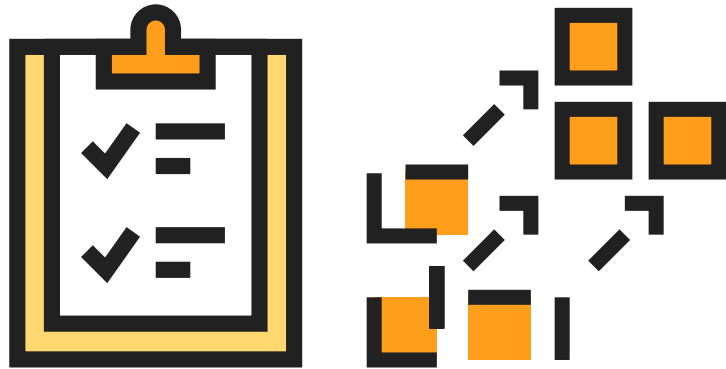
“I would definitely say Puppet was the easiest to ROI out of any investment I’ve had in the last 10 years. Buying a scanning or storage solution, it’s hard to quantify the actual value that goes back to the organization. But with Puppet and how we track our changes, I can relate every change to some amount of time saved, and feed that right back to the executive staff.”

Marcus Vaughan
Director of cloud and enterprise services, phoenixNAP

Step 2: Standardization

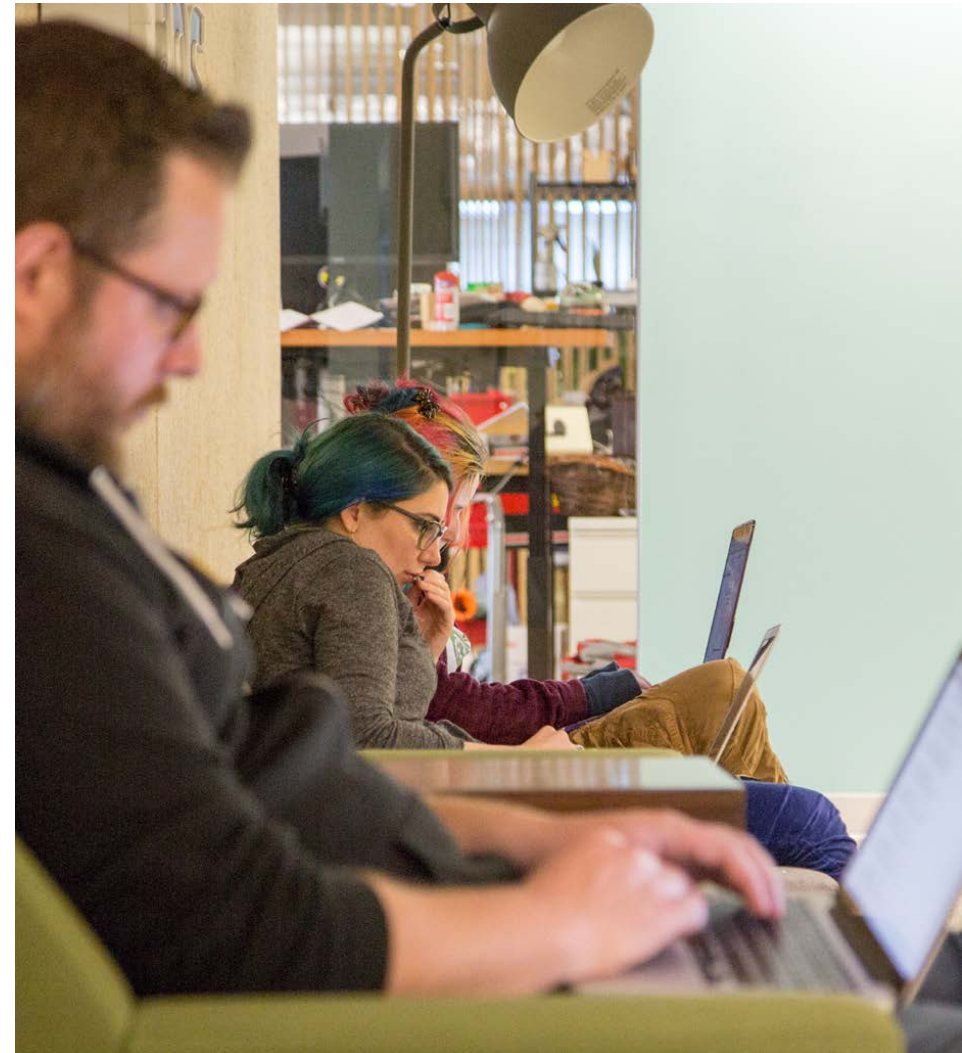
The discovery stage probably showed you just how many different processes and tools are being used across your organization — and how much time is being spent on handoffs and manual integrations.

Now it's time to get your teams working together on a standard set of tools and processes.



Start by deciding which software packaging systems (RPM, Deb, Chocolatey, Docker image, etc.) or artifact repositories you'll use. From there, you'll determine who needs access to specific controls, and who controls access to tools and services. For example, if the development team needs more memory, can they provision it themselves? Do they submit a request? Who reviews and approves the request?

Don't forget the other departments involved in deployment. You'll want to check in with your security team about how secrets are managed — whether there's an agreed-upon way to manage secrets, or whether each team is doing it a different way. You'll want to talk with quality assurance about infrastructure-related issues they run into.



Standardizing certain things first can build a great foundation for the rest of your efforts, depending on how your organization works now. Individuals or cross-functional teams can get started with any of these; all are important:

- Automatically package software, and store packages in a central repository.
- Make build status and progress visible to everyone.
- Include operations in development sprint planning.
- Give developers the ability to provision hosts that include base operating systems, and are already compliant with security and operations controls.
- Provision base configurations (standard operating environment).
- Provision application-specific requirements.
- Provision the entire application.
- Centrally store environmental differences as variables (DNS, NTP servers, etc.) so engineers can deploy to any environment.
- Plan for a phased configuration rollout across environments, starting with a sandbox to test changes, then moving to dev, test, pre-production, production, etc.
- Implement a process for testing infrastructure code. Start with basic syntax validation and linting, then move on to unit testing, and finally to integration and acceptance testing, depending on the sophistication of the team.
- Mirror production monitoring in the test environment to ensure that infrastructure components don't cause test failures. *Note: This is not a substitute for automated testing.*
- Establish an internal process for other teams that need to interact with infrastructure code. Ideally, everyone should have access to the code base, and only a few knowledgeable people have rights to merge changes.



Test-driven development

Test-driven development is key to building a robust DevOps practice in any organization. It means creating tests before writing a single line of code. That may sound counterintuitive, but it's a great way to clarify the intent of the application you're writing.

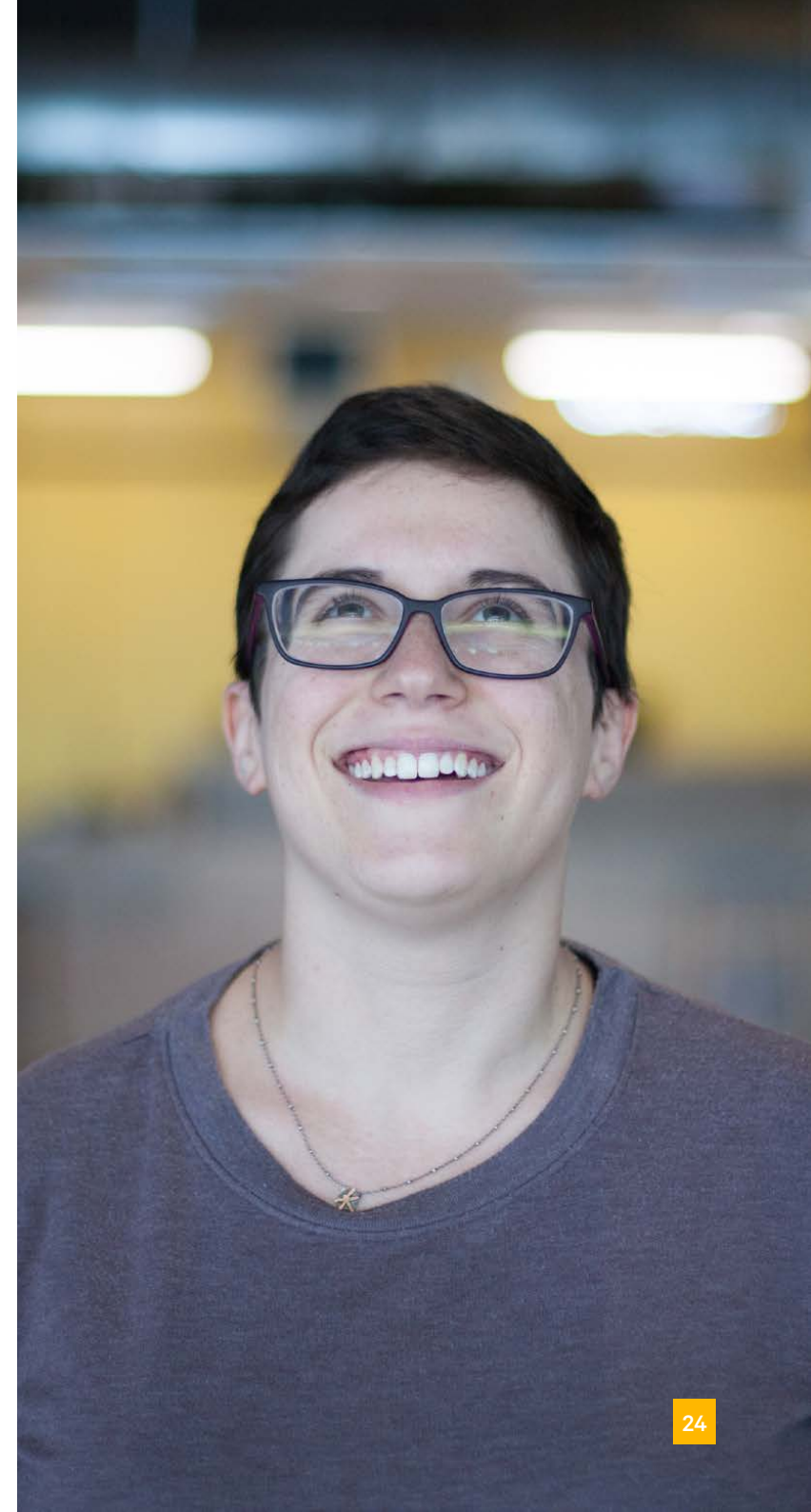
When your infrastructure is code, testing becomes even more important. With Puppet, developers often start with low-level tests, then write code to pass those tests. But where do you start?

Gareth Rushgrove, a senior software engineer at Puppet, recommends writing tests that map one-to-one to Puppet code. "It may not seem that useful," he says, "but it familiarizes you with the testing syntax and the Puppet testing suite." This same approach can be taken with any language.

Should you *always* start with a test? Test-driven development isn't always appropriate. Testing is, but the order in which you create tests depends on the task. If you're working on a straightforward, obvious task, testing after the fact is more valuable. But if you're working on a complex task, or aren't exactly sure about the design, test-driven development can help you define and direct your efforts. It forces you to think from a user's perspective, which may help uncover new needs or features for the software you're creating.

Test-driven development can help you stay on track when tackling large projects. For example, when Puppet engineers were creating a Debian module for the Forge, they could simply run the basic Linux module against a Debian test. The test would generate a list of failures and problems that needed to be addressed. Then engineers could write code to address each failure, until the module completely passed for the Debian use case.

Once you've become comfortable with test-driven development, you can start to automate many low-level tests to help speed up your process. The result will be cleaner, better code.





Step 3: Automation

Manual processes are the enemy of faster delivery. They also wear out your engineers, leaving them tired, depleted, and unable to utilize their full creativity. So automating manual processes is key to DevOps, and to delivering better software, faster.

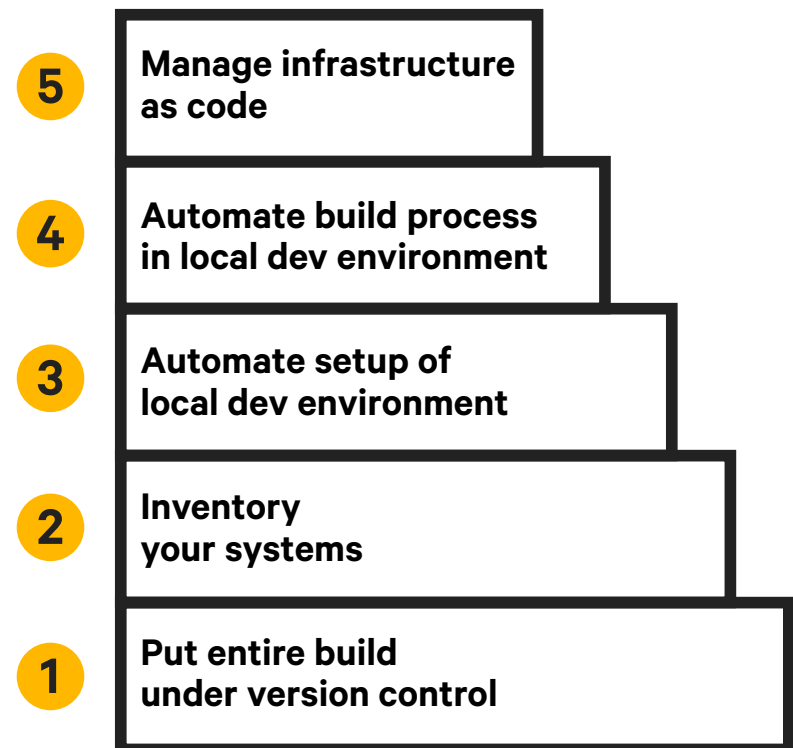
There are other benefits to automation. Automated, repeatable processes are easier to understand, easier to audit, easier to secure, and easier to improve. Automation lets you learn faster and respond more quickly to customer and market demands.

Version control and configuration management are a great place to start. Version control gives you a focal point for workflows and automation, with clear tracking of changes and full auditability. Version control not only enables faster deployment, it also lets you easily roll back to the last good known state when a failure occurs, reducing downtime.

Your first step should be to get everything you need to produce a build under version control — the application code, infrastructure code, deployment scripts, test scripts, environment provisioning, third-party libraries, database schema, etc.

It's a good idea to inventory your systems, and to create a method for organizing and classifying your servers according to their technical or business function (application servers, database servers, ERP servers, etc.). Include details about the different configurations of services — for example, App A and App B may require different Tomcat versions and configurations. This will help you focus on both the business and technical needs in your organization, and draw direct relationships between them. That in turn can help you understand what introducing DevOps processes can do for your business.

5 Steps to Automation





Next, automate the setup of local development environments so they're consistent with your testing and production environments. Create the building blocks of your server environments, including:

- Baseline OS configurations
- Core services: NTP, SSH, DNS, sudo, firewall, task scheduler
- Middleware configurations: web, app and database servers

Then, automate the build process (creation of binaries, packages, etc.) in local development environments. This allows devs to test locally, and changes to be committed only if the build is successful and all tests pass. Automated testing helps transform quality assurance from a pre-production bottleneck to an essential element of software creation, infused at the earliest stages.

“Automation is the best documentation. Gone are the complex and out-of-date documents explaining how to create and install systems. They've been replaced by code which is used to automatically and repeatably create our critical IT systems.”

Geoff Clitheroe, Systems development manager for GeoNet, New Zealand's national geological event alerting service

Finally, and most importantly, manage your infrastructure as code.

This will remove bottlenecks in your service delivery pipeline by enabling better collaboration, as well as enabling automation of your infrastructure. When you treat your infrastructure as code, it's easy to:

- Share and collaborate around infrastructure.
- Put infrastructure into a version control system, just like applications.
- Quickly do peer review, rather than waiting for the next change approval board meeting.
- Perform unit testing.
- Automate deployments.
- And much more.



“We do infrastructure as code. Imagine all your documentation is executable; every step is checked into version control; everything is hyper-consistent, and the tenth stack is identical to the first stack. It starts with the infrastructure, and provisioning that base layer. On top of that, you have your systems, the state and the configuration of the systems. Puppet flushes out that space so that your applications have a very consistent, reproducible on-time infrastructure on which to run.”

A photograph of three people in a meeting room. On the left, a man with a goatee in a plaid shirt looks towards the center. In the middle, a woman with blonde hair in a grey t-shirt looks towards the right. On the right, a man with a beard and glasses in a dark shirt is writing on a whiteboard with a yellow marker. The whiteboard has some green and blue markings. In the background, there are shelves with a sign that says 'PUPPET ENTERPRISE' and some papers. The overall scene is a collaborative work environment.

Step 4: Continuous improvement

Now that you've standardized a good chunk of your technical processes, you have a solid foundation for iterative improvement.

Here's where you stop, take a breath, evaluate where your organization stands now and what can be made better. You're going to measure to see what has improved, and solicit feedback from your teams on what should be improved.

Some questions to ask at this stage

Which pieces of the deployment process are still being done manually? Examples to consider:

- Database migrations
- Setting network firewall rules
- Configuring load balancers
- Provisioning and configuring storage
- Complex application configurations
- Asset management or configuration management database (CMDB) entries
- Production security and compliance reports
- File system expansion
- Scaling and high availability configurations

Which testing process steps are still being done manually?

How successful are deployments, as measured by error rates, service outages, unplanned work, or other quality metrics?

Which additional metrics are we trying to improve?

For example: uptime; deployment frequency; cost per deployment; change lead time; resources managed.

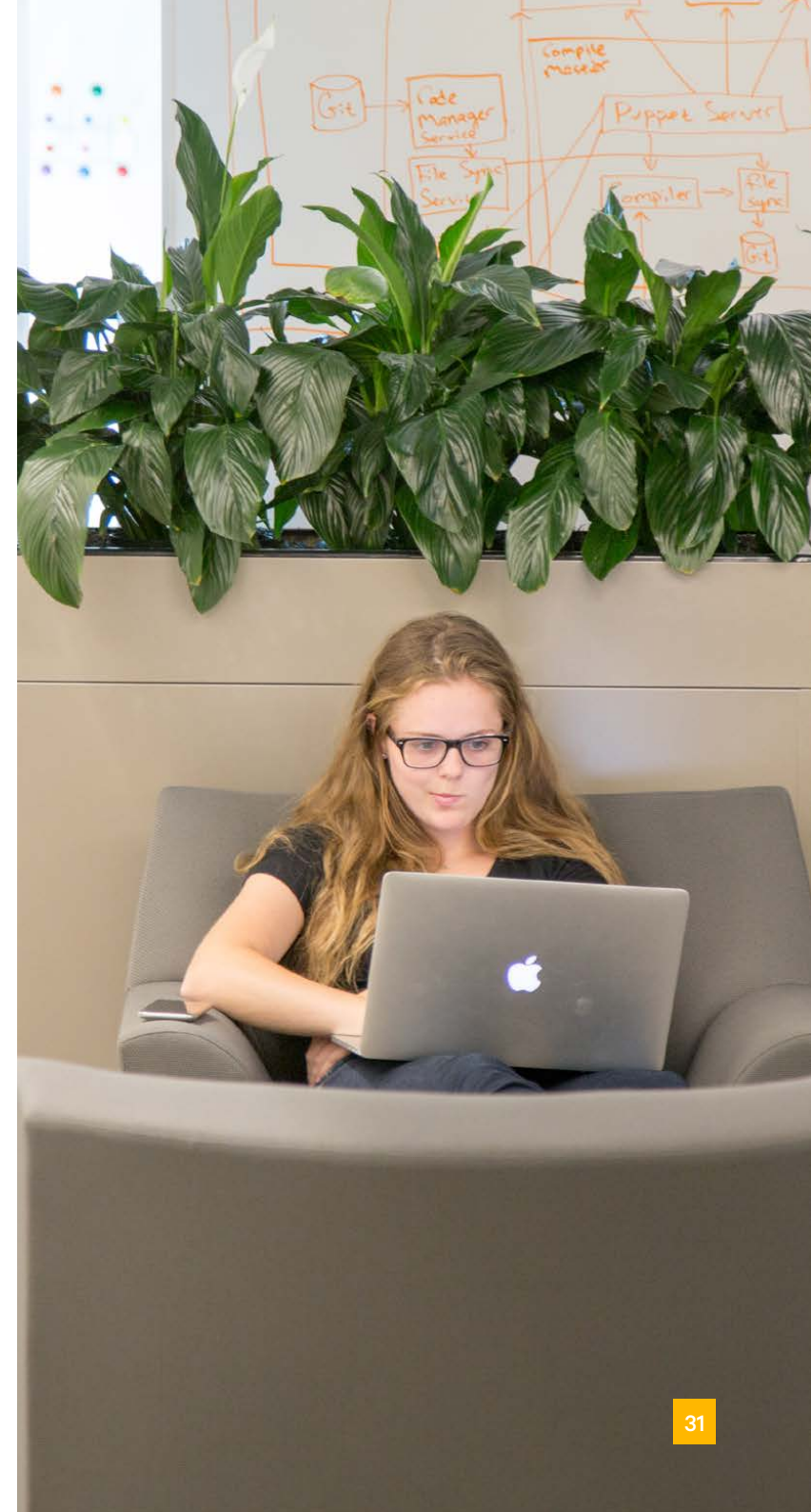
Which tools in our toolchain can be integrated? For example, you could integrate deployment and chat tools so you can run tasks and receive notifications in chat. You definitely should integrate monitoring and configuration management tools.

“I look at DevOps as bringing operations and development closer together to solve business problems, regardless of whether that’s a software release or you have a priority incident to address. I really do see Puppet as a DevOps tool, because it helps bridge that gap between development and operations, and it really level-sets. [For example,] Puppet provides enablement for our development teams to do some work that otherwise they’d be restricted from, from an access standpoint.”

Tom Sabin, IT manager for cloud and automation at Staples

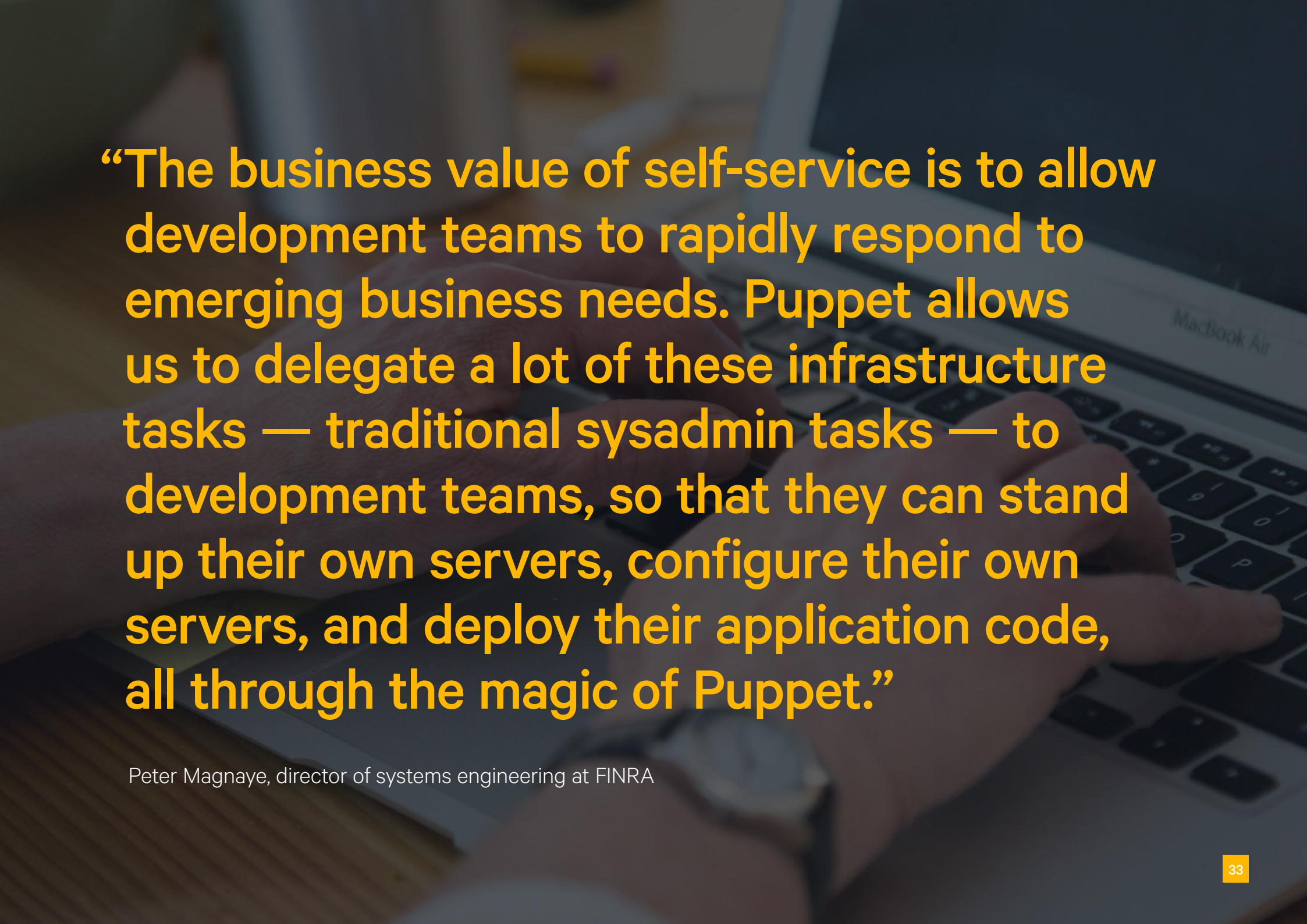
Once you've identified areas for improvement, next steps can include:

- **Automating more of the application deployment process.**
Try automating configuration of more complex applications.
- **Automating more of the infrastructure deployment process.**
This could include automating operational tasks such as adding monitoring checks, adding services to load balancers, recording deployments for audit purposes, etc.
- **Creating opportunities for cross-learning.**
You could hold internal DevOps workshops and hackathons, create cross-functional teams, or institute blameless postmortems — there are many ways to promote cross-learning.





Step 5:
**Driving innovation
and enabling
the business**



“The business value of self-service is to allow development teams to rapidly respond to emerging business needs. Puppet allows us to delegate a lot of these infrastructure tasks — traditional sysadmin tasks — to development teams, so that they can stand up their own servers, configure their own servers, and deploy their application code, all through the magic of Puppet.”

Peter Magnaye, director of systems engineering at FINRA

You've automated the most critical services and applications and established better collaboration practices between the teams involved with deployment.

Now you have a solid foundation for tackling new strategic initiatives that will benefit the business.

The following pages explore some of the most common initiatives that drive management teams to adopt DevOps.

- **Migrating to the cloud**
- **Creating a self-service portal for developers**
- **Automating legacy environments**
- **Adopting new technologies like containers or microservices**



1-800-Flowers.com needed to scale its infrastructure to meet seasonal demand. Holidays like Valentine’s Day and Mother’s Day trigger massive spikes in web traffic and server demand for the gift company, and maintaining a massive permanent infrastructure capable of handling these spikes was just too expensive. The only rational choice was to move to a cloud platform where the company could scale up quickly as needed, and just as quickly, scale back down. So 1-800-Flowers.com rebuilt its entire application stack for the cloud, using DevOps practices.

Transitioning to the cloud is no trivial task, and you can't do it without carefully designed automation. “If we were to use a manual process, and build these machines with the regular resources, we could not ramp up capacity quickly to handle the load we anticipate,” said Veerakishore Vellanki, director of IT infrastructure at 1-800-Flowers.com. “There is too much integration involved; it’s too complex, and you have to tie in with different services. It’s not just one app — there’s a bundle.” The team relied heavily on automation with Puppet to accomplish its cloud migration.

- **Migrating to the cloud**
- **Creating a self-service portal for developers**
- **Automating legacy environments**
- **Adopting new technologies like containers or microservices**



Staples created self-service portals for developers by implementing an internal private cloud. The company uses Puppet Enterprise to standardize and manage configurations, which allows developers to spin up their own machines at will for testing. Staples' IT team has created a user interface that allows their colleagues to order new VMs equipped exactly as they need them, so they can start working right away.

“We’re not just handing out servers; we’re handing out servers with middleware or database on top of it, moving up the stack,” said Jeff Quaintance, senior cloud automation engineer at Staples. “If we have the package already developed, **what took days before now takes literally minutes**. For a new capability — say, a new application server container we need to install — what took several weeks is now down to a week.”

Development cycles are now much faster, making it much easier and less costly for Staples to try out ideas, learn and improve.

- Migrating to the cloud
- **Creating a self-service portal for developers**
- Automating legacy environments
- Adopting new technologies like containers or microservices



Walmart brought more than 49,000 legacy Linux servers in its stores under Puppet management, using DevOps practices such as standardized configuration and automated deployment. The IT team has recently been "puppetizing" the company's fleet of Windows servers, bringing the eventual total number of servers managed by Puppet to at least 100,000. These servers are distributed across 11,000 stores in 27 countries around the world.



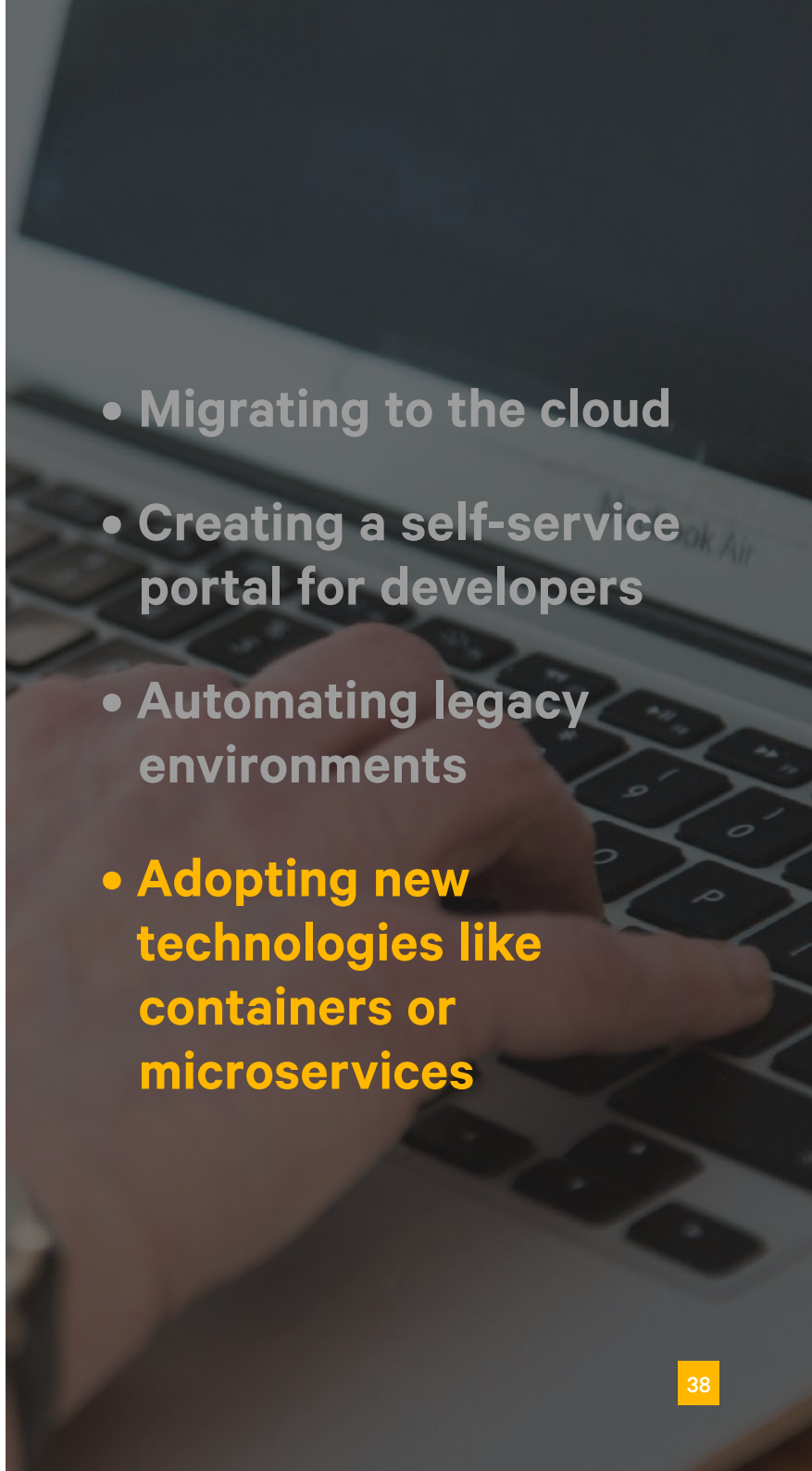
Energy services company **Ambit Energy** runs its infrastructure and applications primarily on Windows and .NET. Before adopting Puppet Enterprise in November 2013, the IT team performed all software deployments manually, and managed about 30 application servers. Using DevOps practices and automation with Puppet, Ambit's IT team now manages about 500 application servers running a variety of services and environments. And the team is still the same size it was before Puppet.

- Migrating to the cloud
- Creating a self-service portal for developers
- Automating legacy environments
- Adopting new technologies like containers or microservices

Traditional ways of managing software and the infrastructure that runs it — the endless meetings, layers of approvals, change ticket backlogs and accumulating cruft — won't cut it in a world where companies run primary applications in the cloud, and try out new technologies like Docker containers, or container-cluster managers like Kubernetes and Mesos.

Your IT organization needs to respond quickly to market changes by making more experiments faster, learning from frequent customer feedback, and then using that learning to rapidly deliver new features. To do that, you need to build your change fitness: your organization's ability to transition gracefully from current practices to new practices, leaving behind old tools and processes as you adopt new ones to serve your business better.

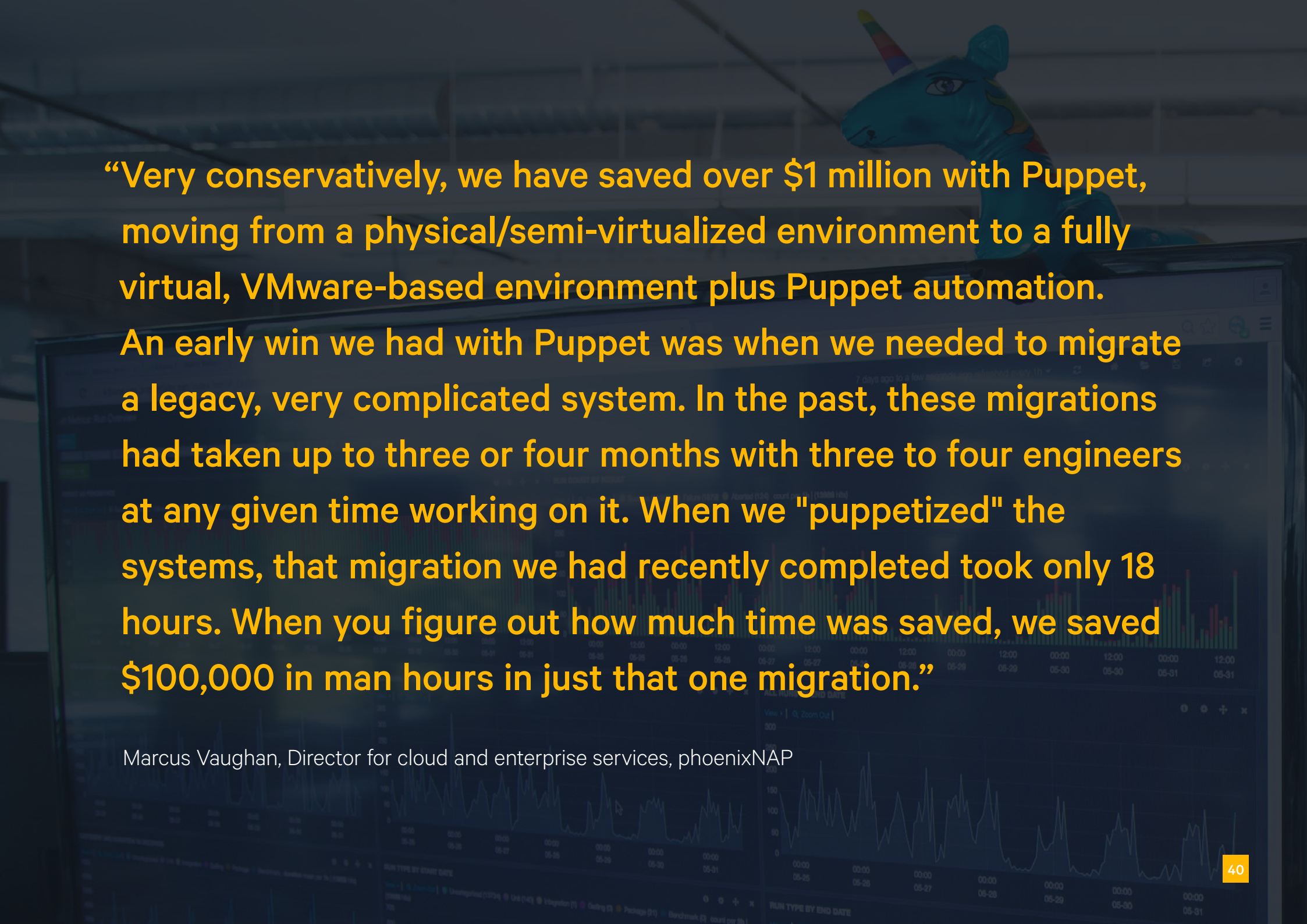
Once you boost your change fitness, IT becomes a streamlined, standardized and forward-thinking organization. Instead of being the last one invited to the meeting, you're leading and driving the business. IT gets to influence the customer experience in a more powerful and direct way than any marketing campaign ever could. Ultimately, there won't be a single project at the company that doesn't rely on IT as a strategic enabler or leader. You'll be prepared to meet digital disruption head-on, and perfectly positioned to create new opportunities for your company, faster.

- 
- Migrating to the cloud
 - Creating a self-service portal for developers
 - Automating legacy environments
 - **Adopting new technologies like containers or microservices**



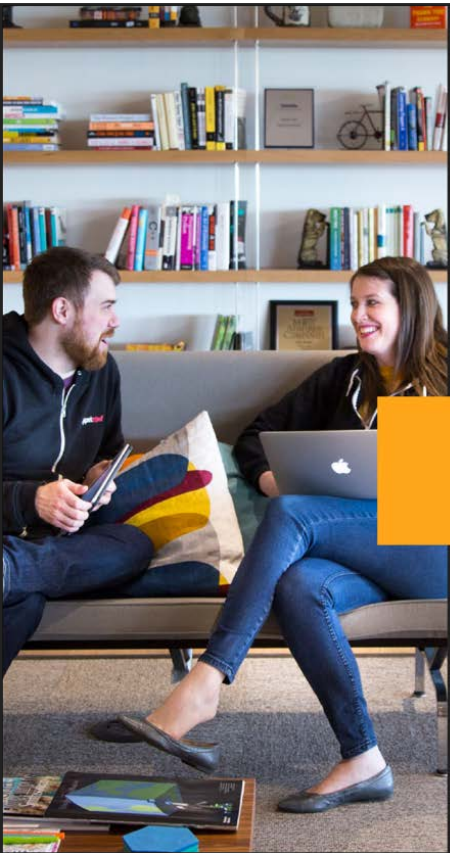
Step 6: Measurement





“Very conservatively, we have saved over \$1 million with Puppet, moving from a physical/semi-virtualized environment to a fully virtual, VMware-based environment plus Puppet automation. An early win we had with Puppet was when we needed to migrate a legacy, very complicated system. In the past, these migrations had taken up to three or four months with three to four engineers at any given time working on it. When we "puppetized" the systems, that migration we had recently completed took only 18 hours. When you figure out how much time was saved, we saved \$100,000 in man hours in just that one migration.”

Marcus Vaughan, Director for cloud and enterprise services, phoenixNAP



2016

State of DevOps Report

Presented by:



Sponsored by:



Any rational tech manager is going to want evidence before they invest their own time — and their team's time — in making significant changes.

That's where the [2016 State of DevOps Report](#) can come in handy. In the fifth annual survey, we asked more than 4,600 development, operations and management people about the practice of DevOps in their organizations.

The results were stunning.

High-performing IT teams — those that use DevOps practices — deploy code 200 times more frequently than their counterparts.

We found that developers who implemented DevOps practices were able to deploy on demand, performing multiple deployments per day. Those who did not rely on DevOps reported deploying between once per month and once every six months.

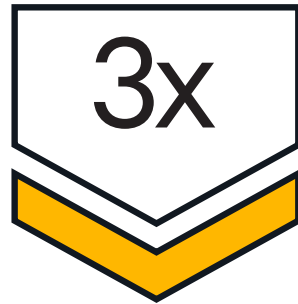
Of course, not every company in every industry needs to deploy thousands of times per day, like Netflix, or even 50 times per day, like crafts marketplace Etsy. But regardless of industry, no ambitious company wants to be held back from deploying as often as makes sense by manual processes, bureaucratic practices or frequent downtime. When you can deploy as often and as easily as you like, suddenly the cost of change becomes much lower. You're freer to experiment, to try things out and learn. Deployment becomes a normal, matter-of-fact process, and your engineers, no longer exhausted by heavy, stressful deployments, have more time to address improvements that can move your business forward.



**200x more frequent
deployments**



**24x faster
recovery from failures**



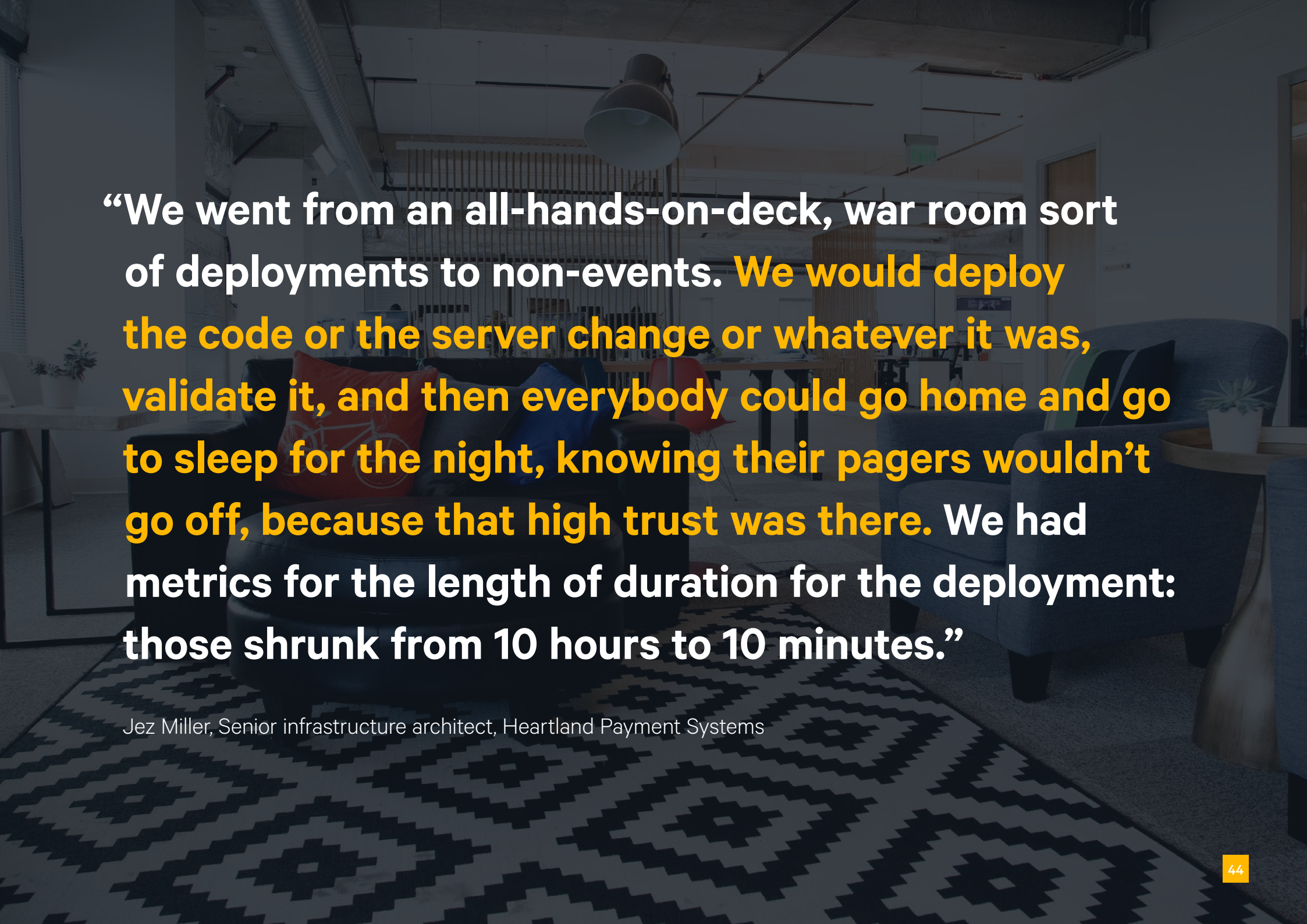
**3x lower
change failure rate**

What's really remarkable is that the increased velocity DevOps shops experience does not come at the expense of reliability. To the contrary: As teams deploy more frequently, the DevOps survey found that they were able to do so with one-third the failures experienced by lower-performing teams. And when failures did occur, high-performing teams that use DevOps practices were able to recover 24 times faster.

Before you embark on your DevOps journey, you will want to make sure you are measuring the things that matter most to your organization: error rates, alert rates, downtime and more. Then at various points along your journey, you can measure again to see how much your organization is improving.

“We are providing a lot of DevOps functionality. We’ve gone from taking 24 to 48 hours to set up a development environment to where a typical server build now takes 20 minutes. There’s a setup for each type of server — A, B or C, whichever flavor the development team needs, they know we will be able to deliver. Every 15 minutes they get back in their day, they can roll into more development work.”

Pope Davis
Senior director of systems engineering at New York Stock Exchange/ICE



“We went from an all-hands-on-deck, war room sort of deployments to non-events. We would deploy the code or the server change or whatever it was, validate it, and then everybody could go home and go to sleep for the night, knowing their pagers wouldn’t go off, because that high trust was there. We had metrics for the length of duration for the deployment: those shrunk from 10 hours to 10 minutes.”

Jez Miller, Senior infrastructure architect, Heartland Payment Systems

DevOps is good for recruitment and retention

Interestingly, successful DevOps practices correlate to greater employee loyalty. As the 2016 State of DevOps survey revealed, employees of high-performing organizations were 2.2 times more likely to recommend their organizations to friends as a great place to work.



Employees in high-performing teams were more likely to recommend their organization as a great place to work.

“It's great that AON cares about the cultural change of DevOps, and invests in the technology to support that change. We have a mature, modern delivery system that most enterprises don't have. I can use that for recruitment.”

Glenn Mason, Solution architect at AON



It costs time and money to recruit and onboard engineers. Once they've learned your company's technologies and business needs, you don't want to lose these people. To keep them, you need to give them the opportunity to work on challenging problems, to use the latest tools, try out new methods and learn. Younger engineers — those raised in a digital world — expect to see results quickly, and are much less accepting of bureaucratic cultures.

The faster release cycles, active collaboration and tighter feedback loops that DevOps enables create a culture of constant learning and improvement. Employees who thrive in that kind of environment are — or soon become — your most productive people. If you don't provide a workplace culture that encourages learning and experimentation, well, there's another employer down the road who does.

Want to know how satisfied your employees are? You don't have to survey them to find out. Take your developers or IT engineers out to lunch; ask them how things are going. People are usually more comfortable discussing points of pain or friction outside the office, in a casual and companionable setting.

These informal meetings can also help you uncover ways to improve the working environment. A quick 10-minute change to infrastructure, for example, might save the dev team hours of work — but until you've heard the root issue, it's unlikely you'll ever understand what needs to be changed. And you need to understand it.

A man in a red long-sleeved shirt is standing in profile, facing right, and writing on a whiteboard with a yellow marker. He is smiling and looking at the board. The background is slightly blurred, showing what appears to be a meeting room or office setting.

Finding time: Let it burn

Finding the time to implement DevOps practices can be difficult when your IT and Dev staff are busy fighting fires. Our advice? Let it burn.

Some problems were there yesterday and will be there tomorrow. If all you're doing is reacting to those problems, you're not improving anything.

So let the non-disastrous issues continue, and use your team's time to automate something that drains their time on a regular basis. If you automate something that takes a few hours a week to do by hand, suddenly you have those hours available to make more improvements. Bootstrap your way from one piece of automation to another in this fashion, and a year from now, your team will be spending a lot more of its time innovating — and the business will be further ahead.

Letting a fire burn also can help reveal what's really needed in your workplace. Maybe you need different people on your team. Maybe you need more people. Maybe you're using some tools that don't really fit your needs. Maybe the stacks or software you're supporting aren't right for your organization.

We can't tell you which fire to let burn. Some are less problematic than others. Perhaps some burn hottest only at the end of the month, or on Tuesdays. So look at your own situation, select an area you can cordon off, and let that fire rage. It might be your first step toward actual progress.





Implementing a solid DevOps practice in your organization will take time.

Revamping and automating some simple tasks can take months, and some, like metrics, may never be completed. Remember, DevOps isn't a task or project you can cross off the list. It's a way of working, a philosophy that will guide your development and IT practices forever.

When you look at the results achieved by DevOps organizations, it's easy to see that DevOps is just a better way to make and distribute software. When you measure your own results, even just a few months into a DevOps transformation, you can see improvements that have a real impact on your business. Whether you're a Fortune 500 conglomerate, a scrappy startup or a non-profit organization, DevOps practices can dramatically improve the way you work, and your company's ability to get ahead of the competition.

So get started now.

Resources

2016 State of DevOps Report

The fifth annual industry-leading survey of IT professionals discusses the state of deployment, security, stability and employee loyalty at organizations that have (or have not) adopted DevOps. The report digests findings from the most recent survey of more than 4,600 professionals, plus learnings distilled from more than 25,000 respondents worldwide over the past five years.

Get Started with DevOps: A Guide for IT Managers

Practical advice for managers of IT teams on how to work effectively and empathetically with their own teams, development teams and upper management.

Hiscox Reduces the Cost of Release by 97% with DevOps and Puppet

A global specialty insurance company details its DevOps journey, including initial steps, how it overcame obstacles, and its plans for spreading DevOps throughout the organization.



About Puppet, Inc.

Puppet is driving the movement to a world of unconstrained software change. Its revolutionary platform is the industry standard for automating the delivery and operation of the software that powers everything around us. More than 33,000 companies — including more than two thirds of the Fortune 100 — use Puppet's open source and commercial solutions to adopt DevOps practices, achieve situational awareness and drive software change with confidence. Based in Portland, Oregon, Puppet is a privately held company with more than 500 employees around the world.

Learn more at puppet.com.