

The 5 Foundational **DevOps Practices**

How to establish and build on them

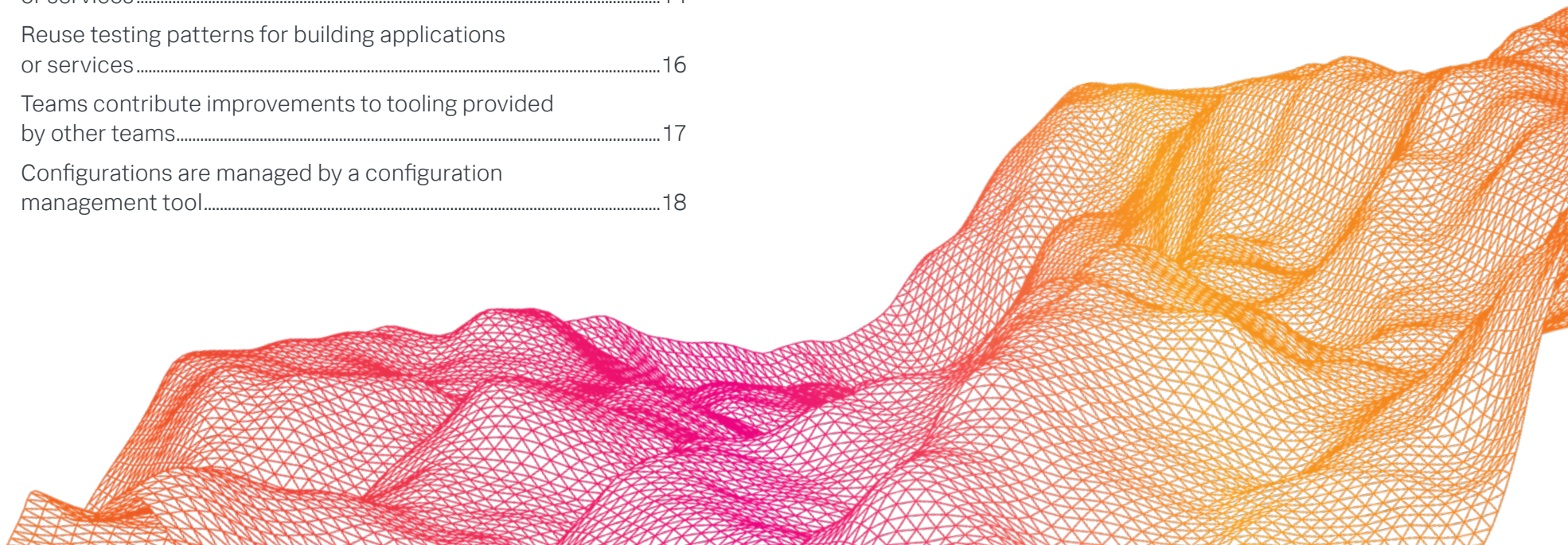


splunk>
turn data into doing

Table of Contents

Introduction	3
Choose your own adventure.....	4
The foundational practices and CAMS	5
It's all about sharing.....	6
The five foundational practices one by one	7
Monitoring and alerting are configurable by the team operating the service.....	8
Reuse deployment patterns for building applications or services.....	14
Reuse testing patterns for building applications or services.....	16
Teams contribute improvements to tooling provided by other teams.....	17
Configurations are managed by a configuration management tool.....	18

Implementing the foundational practices in your organization.....	20
But ... start where you are.....	21
The best path is the path that will show quick results	22
About the author	23





Introduction

It's taken less than a decade for DevOps to move from hot new buzzword to being widely acknowledged as the right way to manage technology change and deliver software in a fast-moving, highly competitive business environment. But even with plenty of examples offered at tech conferences around the world, in books and on leading blogs, most teams still struggle with how to get started, and where to go after these first steps.

We've researched how organizations progress through their DevOps journeys, and after analyzing the data, we found that the successful ones go through specific stages. Our research also revealed a set of core practices — we call them “foundational practices” — that are critical to success throughout the entire DevOps evolution.

Choose your own adventure

Every organization starts from its own unique place. It has legacy technologies, established ways of doing things, its own specific business mission and its own particular culture. So there is no single path to a DevOps transformation; instead, there are many possible evolutionary paths.

Analysis of the data from the 2021 State of Observability report revealed the foundational practices that successful teams employ. These practices correlate so strongly with DevOps success, we've determined they are essential at every stage of DevOps development. In other words, the practices that must be adopted at any given stage in order to progress to the next stage remain important even for those organizations that have evolved the furthest on their DevOps journey, and that have already showed the most success.



Each foundational practice can be described in a sentence:

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

When we examined each of these practices more closely, we found that highly evolved organizations were much more likely to always be using these practices throughout the evolutionary journey than the less-evolved organizations. What we take from our findings is that the foundational practices listed above are integral to DevOps, and critical for DevOps success.

The foundational practices and CAMS

The importance of the foundational elements of DevOps shouldn't surprise anyone who takes more than a passing interest in DevOps. Other well-regarded constructs are built around these same foundations. The CAMS model,¹ one of the earliest descriptions of DevOps, encompasses these foundational known-good patterns, from the importance of measurement and sharing to the need for automation. Other tropes and methodologies common in the DevOps discourse — concepts such as shifting left, empowered teams, test-driven development and more — also reinforce these foundational patterns and practices.

Culture
Automation
Measurement
Sharing

¹ Culture, Automation, Measurement, Sharing. CAMS was first defined by Damon Edwards and John Willis in 2010. For a longer discussion of CAMS and DevOps, see "CAMS and the DevOps evolutionary model," a chapter in the [Puppet 2018 State of DevOps Report](#).

It's all about sharing

On studying the foundational practices, we realized that they are all dependent on sharing, and that they all promote sharing. When considering what is shared in a DevOps practice, we start with configuration and monitoring/alerting.

Monitoring and alerting are:

- Configurable by the team operating the service.
- Key to sharing information about how systems and applications are running.
- Designed to get everyone to a common understanding of how systems should be running.

This common understanding is vital for making improvements, whether within a single team and function or across multiple teams and functions. Other sharing patterns include:

Deployment patterns and testing patterns for building applications or services are reused. Sharing successful patterns across different applications or services often means sharing across different teams, establishing agreed-upon ways of working that provide a foundation for further improvements.

Teams contribute improvements to tooling provided by other teams. This form of sharing promotes more discussion between teams around priorities and plans for further improvements in tooling, process and measurement.

Configurations are managed by a configuration management tool. A configuration management tool enables development, security and other teams outside Ops to contribute changes to system and application configurations. This makes operability and security a shared responsibility across the business.

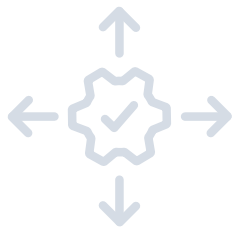
Our discovery that all the fundamental practices enable or rely on sharing tells us that the key to scaling DevOps success is adoption of practices that promote sharing.

It makes sense: When people see something that's going well, they want to replicate that success, and of course people want to share their successes. Let's say your team has successfully deployed an application 10 times, and let's also say this type of deployment has normally given your team (and others) a lot of trouble. Chances are, someone will notice and want to know how you're doing it. That's how DevOps practices begin to expand across multiple teams.



The five foundational practices one by one

In this section, we describe each of the foundational practices in some detail, and how the practice contributes to the evolution of DevOps.





Monitoring and alerting are configurable by the team operating the service

Core to the DevOps movement is the two-sided coin of empowerment and accountability, which Amazon CTO Werner Vogels summarized in his famous statement: “You build it, you run it.”² So our research looked at how many teams that run applications and services in production — whether comprised of devs, operators, software release engineers or others — are able to define their own monitoring and alerting criteria.

Empowered teams that run applications and services in production can define what a good service is; how to determine whether it's operating properly; and how they'll find out when it's not. This empowered monitoring approach can take many forms. For example:

- “Drop a monitoring config in a location and we'll pick it up.”
- “Log into this web interface to configure your monitoring.”
- “Add some monitorable outputs to your infrastructure code.”
- “Here's an API for you to configure monitoring as code.”

We found that once organizations start to see traction with DevOps, they are able to define their own monitoring and alerting criteria for apps and services in production. Empowered teams that run applications and services in production can define what a good service is; how to determine whether it's operating properly; and how they'll find out when it's not. They also leverage automation to multiply the effects of their work and to minimize toil (repetitive tasks that add little value).

² Gray, J., Vogels, W., A Conversation with Werner Vogels, ACM Queue, <https://queue.acm.org/detail.cfm?id=1142065>, June 2006, retrieved Aug 2018.

Empowering teams to define, manage, and share their own measurement and alerting supports multiple elements of a DevOps transformation, including:

- Sharing metrics as a way to promote for continuous improvement
- Creating and promoting a culture of continuous learning
- Cross-team collaboration and empowered teams
- Development of systems thinking in individuals and teams

These factors are core to a strong DevOps culture, as we discussed earlier, so it's not surprising that the highly evolved organizations we surveyed adopt this practice early.

What to measure — and how

Our research showed what to monitor and alert for:

- **Key system metrics** such as latency, response time, resource utilization and more. A majority of respondents (56 percent) expose these.
- **Business objectives** derived from system metrics for example, measuring response time as a proxy for customer satisfaction. More than a third of respondents (37 percent) monitor these.
- **On-demand access** to real business metrics such as time on site, application opens, customer sign-ups, revenue rates, etc. Twenty percent of respondents provide this.

Collecting this data provides you with the best opportunity for analyzing and figuring out where problems have occurred in modern, complex systems. Having all of this data available makes it easier for anyone to determine where issues have occurred and to fix them faster.



Importance of monitoring real business metrics

Our research on the State of Observability revealed that missing issues causes problems; 45% of respondents reported lower customer satisfaction and 37% reported loss of revenue as a result of service outages. Measuring business metrics helps you catch these issues before have impact to your business.

Some IT teams use business metrics as primary measures of IT performance.

Business metrics answer day-to-day service questions such as:

- Can customers buy from us?
- How many people are we serving?
- How much money are we making?
- Is this pattern normal?

Business metrics are also important for long-term measures of success such as:

- Does the new release do what we expected?
- Does it drive better business outcomes?
- How does its business impact compare to other versions?

For more advanced organizations using data analytics, machine learning, and artificial intelligence, business data provides early warning of failure by alerting to changes such as lower revenue per minute and a higher-than-usual bounce rate that could be caused by undetected website errors.

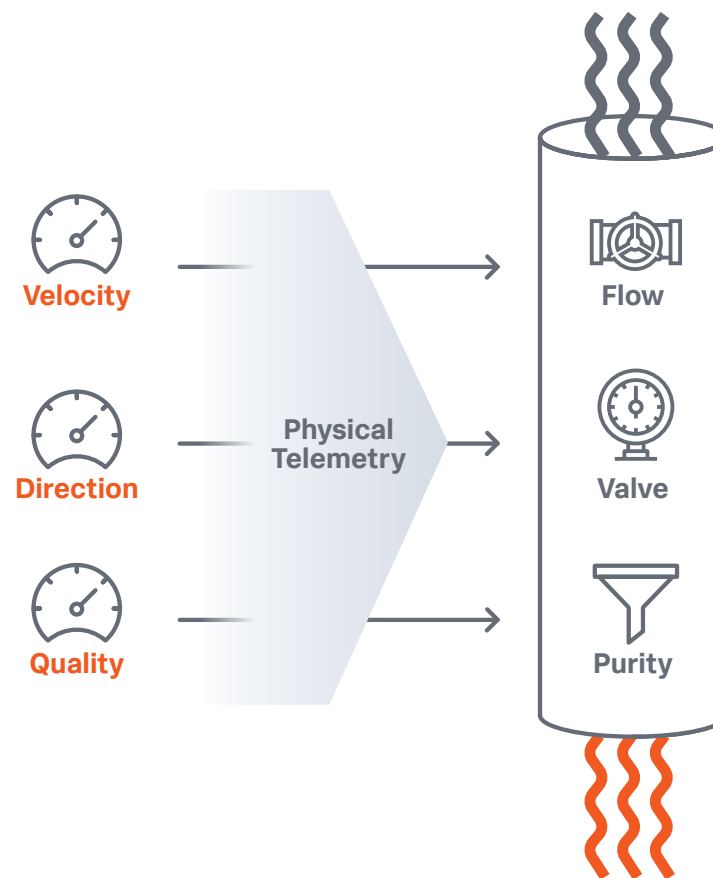
Monitoring business metrics can also help a team get more resources for its DevOps initiative by demonstrating the successes achieved so far in terms the business can understand.

On observability, telemetry and semantic logging

Among highly evolved organizations, the majority always deploy application metrics, traces, and logging along with the application or service, so that monitoring and alerting are simplified downstream in production operations. This provides “observability”: the ability to answer questions about your business, including ones you didn’t know you would have in advance.

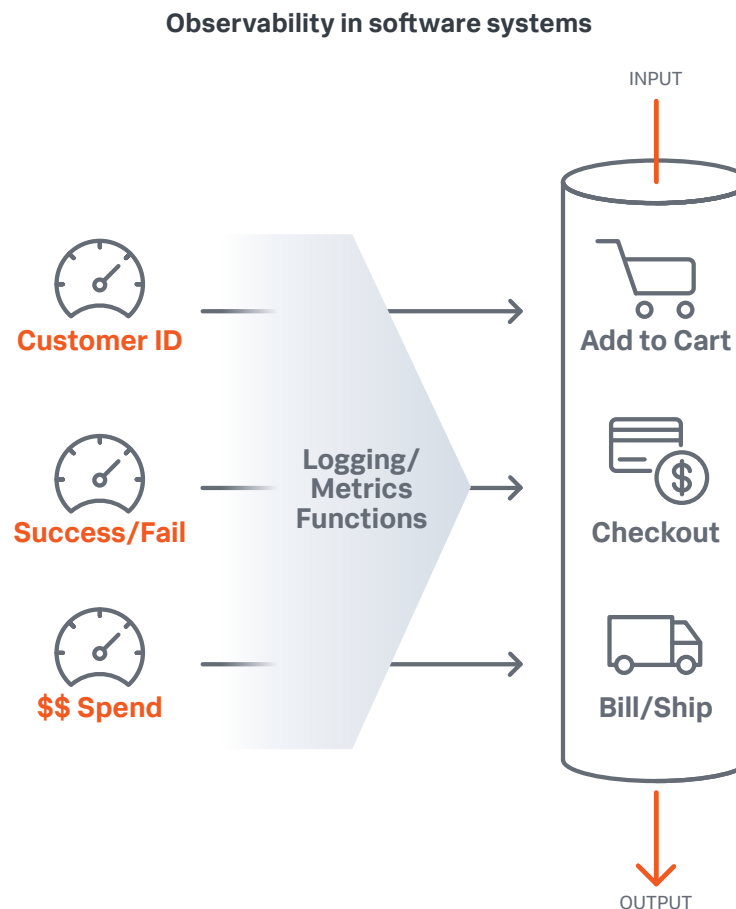
The term “observability” originated in industrial systems, and achieving observability can be as simple as adding a flow gauge inside a water pipe (i.e., an internal measurement), and connecting it to an external display (i.e., adding telemetry). Doing this allows an operator to observe an internal property of the system (how fast water is flowing inside a pipe) by observing an external output (the flow meter mounted outside the pipe).

Observability in industrial systems



In an application, observability is often achieved by adding performance metrics, microservice call tracing, and semantic logging to describe the real activity of an application, which can include both technical metrics such as end-to-end response time and business metrics such as revenue per transaction. This enables teams to configure their monitoring using actual measures of success — both technical and business success — rather than having to guess at results.

Observability enables teams to work with modern composable architectures such as the cloud, microservices, containers, APIs and serverless infrastructure. For all their advantages, these architectures may provide little visibility into infrastructure performance, no ability to inject instrumentation (and minimal native instrumentation), and no way to execute synthetic transactions. In such complex, opaque systems, emitting metrics, traces, and logs to build observability into the system at release time puts DevOps practitioners in the driver's seat, able to define and configure meaningful monitoring for any future scenario.



Measurement is a core piece of DevOps

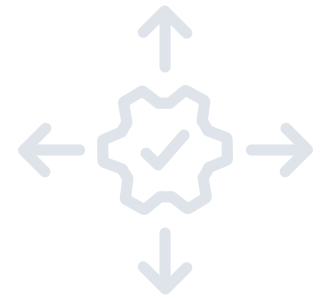
Empowered monitoring isn't for just Ops or for some newly created DevOps team: It's for all teams that work with technology. The embrace of empowered monitoring for all teams underlines one of the most important points of DevOps: that you don't need to create a new team with new superpowers, but instead should empower all existing teams so they can work together in new ways.

Self-service monitoring and alerting can just as easily and usefully be adopted by:

- Developers monitoring their own code as it runs in production.
- A team of developers working with their operations counterparts to deliver operable applications.
- A DevOps team working as a single cohesive group to define their own monitoring practice.
- A complex team of teams where ops specialists monitor applications delivered by devs as part of a broader system.

Regardless of the specific circumstances, self-service monitoring and alerting is a countermeasure to the long-standing anti-pattern of Dev and Ops working in silos. Simply opening access to these key metrics enables a sharing culture, populates feedback loops and promotes a culture of continuous learning across teams.

02



Reuse deployment patterns for building applications or services

Our research reveals a second baseline capability that is critical to DevOps success: the ability to reuse pre-defined deployment routines, processes, systems and tools for building either applications or entire end-to-end services.

Reuse of code is essential to high-quality software engineering, and this applies to deployment code and monitoring code as well. Use of monitoring-as-code and infrastructure-as-code speeds up release cycles, reduces mistakes, and further empowers both Dev and Ops to do their best work.

In a recent survey conducted by Puppet, they asked, “How frequently were these practices used after you started to see some traction with DevOps?” Here is the breakdown of responses for the practice, “We reuse deployment patterns for building applications or services.”

	LOW	MEDIUM	HIGH
ALWAYS	2%	14%	46%
MOST OF THE TIME	7%	44%	47%
SOMETIMES	34%	33%	6%
RARELY	40%	8%	1%
NEVER	18%	1%	—

We asked, “How frequently were the following used while you were expanding DevOps practices?” Here is the breakdown of responses for the practices, “We reuse deployment patterns for building applications or services.”

	LOW	MEDIUM	HIGH
ALWAYS	—	11%	52%
MOST OF THE TIME	7%	45%	47%
SOMETIMES	30%	35%	1%
RARELY	45%	8%	—
NEVER	18%	1%	—

We asked about reuse of deployment patterns because of the special nature of application deployment in most, if not all, organizations. Residing at the boundary between development and production, application deployment is where Dev and Ops most often meet — and most painfully collide. So improving application deployment is right at the core of DevOps, as it mediates the “wall of confusion”³ at the intersection of Dev and Ops.

The use of repeatable patterns — whether created in-house or adopted from an external source — does more than alleviate the immediate pain and confusion of deployment. It also makes it possible to share and spread the knowledge of how to deploy more widely in the organization, enabling more teams and individuals to work together on what needs to be a core competency for any business.

³ <http://dev2ops.org/2010/02/what-is-devops/>

03



Reuse testing patterns for building applications or services

Automating the testing of your software is as important, if not more so, than automating its deployment and monitoring. Small changes can interact with other services in unpredictable ways, and without a robust testing system in place, you could miss those changes.

Here are some considerations as you prioritize this practice:

- **If quality teams are quite disconnected from Dev and Ops teams, you may want to wait to integrate them into DevOps initiatives later on.** Focus on establishing good testing patterns within your own team first. For ops teams, that could mean having a process for testing infrastructure changes before deploying to production. For dev teams, that could mean implementing test-driven development (TDD) or other methodology as part of your agile workflow.
- **Activities closest to production, such as provisioning, monitoring, alerting, etc., are often higher priority for teams because that's where more issues become visible.** Solve your deployment pains first to gain back time you can then use for improving your testing practices.
- **Testing patterns may be less reusable than deployment patterns** because testing deals with the specifics of an application or service, and also covers many different processes — smoke tests, unit tests, functional tests, compliance tests, complexity tests — in both static/white box or dynamic/black box environments.
- **Testing in production is harder and often more complex than testing in pre-production,** as its goals are different from those of a pre-production test. With continuous delivery, teams can experiment in production to test new ideas (e.g., via blue/green or canary releases). This is valuable, but it's different yet again from testing in production, which focuses on customer experience, quality, functionality, resilience, stability and more..
- **The adoption of reusable test processes is a fundamental practice but tends to get pushed out later in the evolutionary journey,** after deployment patterns are well established. If you have to prioritize, we recommend waiting to tackle this one and focusing on the other practices first. However, once you do adopt this practice, it's important to ensure that testing patterns are shareable. For example, you can encode reusable tests into automated testing tools, and share access to those tools — along with the resulting reports or dashboards — among all stakeholders.

04



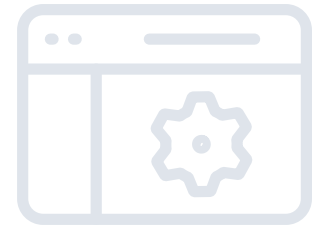
Teams contribute improvements to tooling provided by other teams

The ability to contribute improvements to tooling provided by other teams stands out as a key foundational capability.

Improvements to tooling are typically manual and ad hoc, and siloed within a single team until some change drives the need to open up to other teams. This change might be division-wide culture or organizational changes; new cross-boundary data sources such as semantic logging; or new collaboration across teams at functional boundaries such as provisioning or release automation.

Because the practice of cross-team contributions to tooling is dependent on teams putting their own houses in order first, we believe adoption of this practice can be left to a later stage with equal chances of success.

05




Configurations are managed by a configuration management tool

The practice of managing configurations with a configuration management tool rapidly takes root once organizations start to see traction with their DevOps evolution.

For long-time DevOps devotees, it is not surprising to see this practice emerge as a baseline for success. Automated configuration management was a prime mover of DevOps for many years, especially in the earliest days of thinking about infrastructure as code, and the DevOps movement largely coalesced

around the earliest innovators in automated configuration management. Lately, moving to the cloud further emphasizes the importance of having a configuration and deployment management system to manage the complexity of a modern deployment.

Looking at the early drivers for DevOps explains the importance of configuration management.



As the consumerization of IT drove an unprecedented demand for speed and service, developers' inability to provision and configure key resources for development and testing — let alone production — became a significant impediment to rapidly providing high quality software that would meet customer and business demands.

Meanwhile, IT ops teams were just as frustrated by their inability to make any substantial changes to the applications and services they were running — changes that would make these apps and services more stable, more efficient, and more operable.

The emergence of cloud computing gave developers the ability to provision for their own needs, just by pulling out a credit card. But this new capability also created more problems and frustration for Ops. As developers resorted to shadow IT, Ops teams were faced with a multitude of uniquely configured, difficult-to-manage, highly brittle and error-prone snowflakes in production.

As developers and operators started working together to solve these difficulties around provisioning, it was natural for many DevOps initiatives to begin with automating configuration and provisioning. It's a solution that allows both Dev and Ops to win, by enabling developers to move at market speed while encapsulating and codifying known-good configurations to reduce the pain for operators.

As DevOps evolves and expands, and developers liberated by automated provisioning move increasingly toward continuous delivery, Ops is under even greater pressure to maintain uptime, performance and availability in production. Auditability concerns also emerge as an organization's processes mature. So automated configuration and provisioning for production become just as important as provisioning for development and testing. Achieving repeatability via configuration management assures stable, reliable and auditable production environments, and also enables later-stage capabilities — for example, self-service — that emerge as new goals for the DevOps initiative.

Finally, the importance of automated provisioning and configuration is not just about speed. It also defends against one of the earliest management fears around DevOps: that empowered developers could go rogue on production systems with no recourse, no auditing, and no control. Codifying known-good processes and executing them through preset routines allays these fears, while still enabling Dev and Ops teams to work with greater agility. Especially for larger organizations and those in regulated industries such as healthcare, finance and government, automation offers the added advantage of maintaining a record showing who touched any system, what they did and when they did it — and often, why.

Implementing the foundational practices in your organization

With so much evidence that the five foundational practices lead to success, we know our readers will want guidance on how to implement them. Fortunately, our research does provide some indication of which practices to start with.

In general, the research suggests organizations should start out with baselines that set the foundation for future growth and development, while also helping to deliver immediate value. Which practice to start with depends very much on where an organization is at the outset, and what its most urgent needs are. But do take note: every one of the foundational practices is adopted early by highly evolved organizations.

Three practices are always in active use by the majority of highly evolved organizations by the time they see some traction in their DevOps initiatives:

- Reusing deployment patterns.
- Using a configuration management tool.
- Allowing a team to configure monitoring and alerting for the service it operates.

While these three practices are foundational capabilities, and form a baseline for progression to higher levels of DevOps evolution, the order in which they are adopted is not important. Start here, but don't worry about which comes first. It's likely you'll recognize one particular practice as something your own organization needs to prioritize.

Our research strongly suggests organizations need to have more of the foundational baselines in place before they start to show results. So, work to adopt the three practices above, and then expand them to other teams as soon as you can.

The remaining two foundational practices are ones we recommend prioritizing after the other three practices are well established:

- Reuse testing patterns for building applications or services.
- Empower teams to contribute improvements to other teams' tooling.

But ... start where you are

Every organization is different. You should start where you are now. Identify the existing conditions in your organizations, and choose the approach that will best show early positive impact where your organization needs it the most. With the fundamental practices offering better monitoring, better configuration control, better deployment patterns, better quality controls, and collaborative tooling, there's plenty of scope for choosing.

Establish and share baseline measurements for system, customer, and business metrics at the outset, so everyone knows where you are starting, knows what needs to improve, and knows the impact of their work on the end-to-end system. This will help everyone on your team begin to embrace DevOps concepts like continuous feedback and continuous improvement.

Share work practices and processes early so you can start to move forward as a team. Automate, and shift left with automation of configuration and provisioning to help developers achieve agility and operators achieve predictability. This will help people on both teams believe in the value of the new venture. Making people's work lives better is a great way to earn their confidence and their continuing cooperation.

A shift left refers to doing something earlier in the process than it was traditionally done. Moving configuration and provisioning planning to the beginning of the process ensures you are dedicating the right amount of attention to it.

The best path is the path that will show quick results

Every organization is different, so start with what matters most to the people whose blessing you need. Choose an effort that will show the kind of success that will win the confidence of leaders in your organization. This will buy you more time, more resources, and more budget to move forward and build on your early success.

[Learn More](#)

After you've started work on this DevOps journey, you will find that you need better tools to figure out the health and performance of your business. Splunk's answer to modern applications is Splunk Observability Cloud.

About the author



Greg Leffler

Greg heads the observability practitioner team at Splunk and is on a mission to spread the good word of observability to the world. Greg's career has taken him from the NOC to SRE to SRE management, with side stops in security and editorial functions. In addition to observability, Greg's professional interests include hiring, training, SRE culture, and operating effective remote teams. Greg holds a master's degree in industrial/organizational psychology from Old Dominion University.



Splunk Inc. turns data into doing with technology that is designed to investigate, monitor, analyze and act on data at any scale.



puppet

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 40,000 companies — including more than 75 percent 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. Headquartered in Portland, Oregon, Puppet is a privately held company with more than 500 employees around the world. Learn more at puppet.com.

Learn More

Splunk, Splunk> and Turn Data Into Doing are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names or trademarks belong to their respective owners. © 2022 Splunk Inc. All rights reserved.

22-22512-Splunk-The 5 Foundational DevOps Practices-LS-116



splunk>
turn data into doing®