# Devops:
# A Cynics Guide

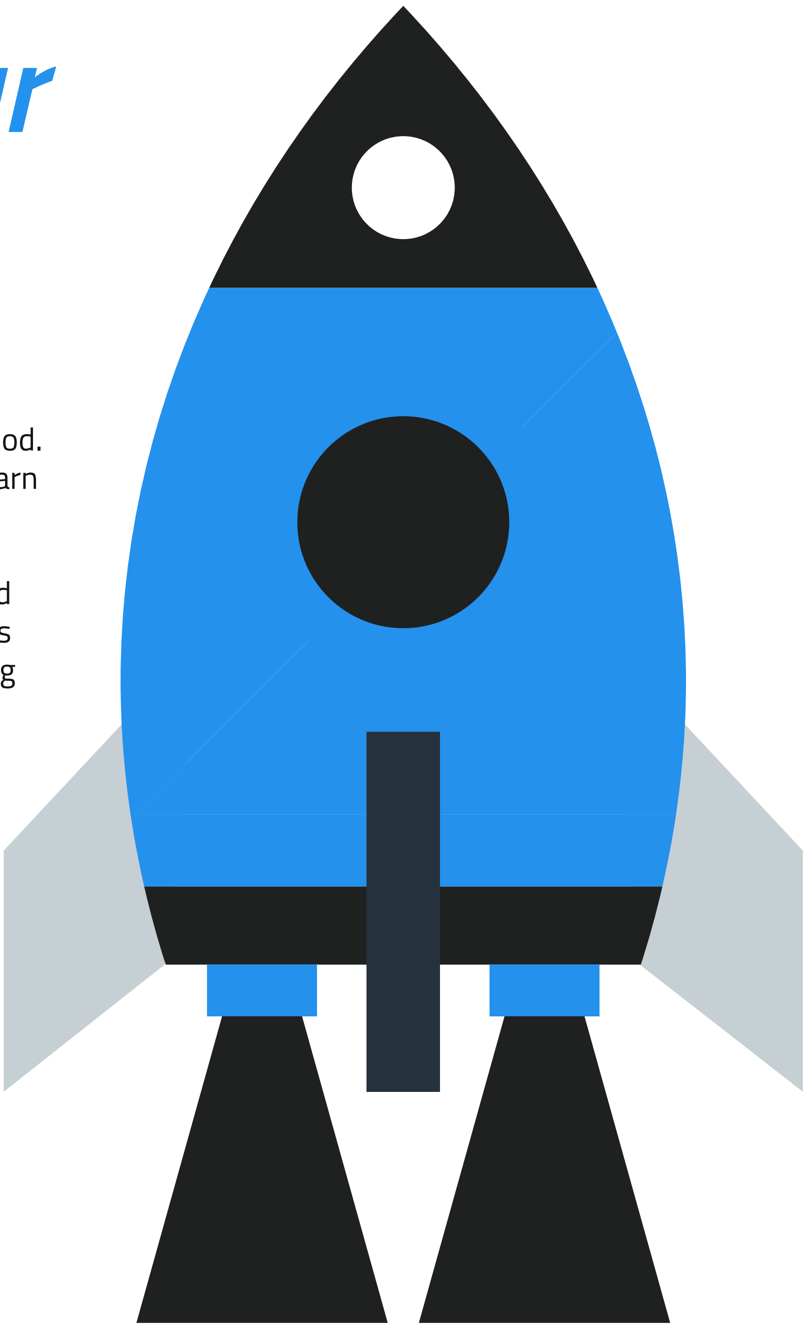
UpGuard

# Table of Contents

# Who is this book for?

You don't have time for bullshit. You know what "measure twice, cut once" means. You want to use the best tools for your job. You stop to think about what you're doing. You care about quality, and planning, and unintended consequences.

## *This isn't your first rodeo.*



You've heard about DevOps. Some of it sounds good. Some of it sounds like bullshit. You're willing to learn more.

This book avoids the DevOps culture wars. Instead of arguing about what DevOps is or should be, this book uses the most common sense understanding of DevOps and talks about what works and what doesn't.

If you think anything named DevOps has to be good, this book isn't for you.

***This is the Cynic's Guide to DevOps.***

## DevOps is a Simple Philosophy

### In theory, DevOps is good for every business.

But theory doesn't always translate perfectly to reality. Theory is an advertisement. Reality is a data set.

The term "DevOps" is about 5 years old, and the community still has no consensus on what that word really means, even though it is overflowing with thought leaders who will claim to be able to tell you.

The core of DevOps is not a difficult or complex concept. More or less, it's the idea that developers and operations should collaborate more closely than they traditionally have, so neither side is caught with their pants down when something blows up.

The core ethos has spawned many valid accoutrements, but at its simplest definition it is about us all "rowing in the same direction."

Developers and Operations, we are all in the same boat. We have to work together to make it go forward.

Creating a "DevOps Team" is completely missing the point – you have just created yet another silo. Instead of two groups trying to coordinate releases, there are three.

The goals of DevOps—or really, the goal of any improvement  to software development—is to ship good code sooner. The "Ops" part is there as a reminder that code isn't good unless it works in Production.

## DevOps Has Mass Appeal

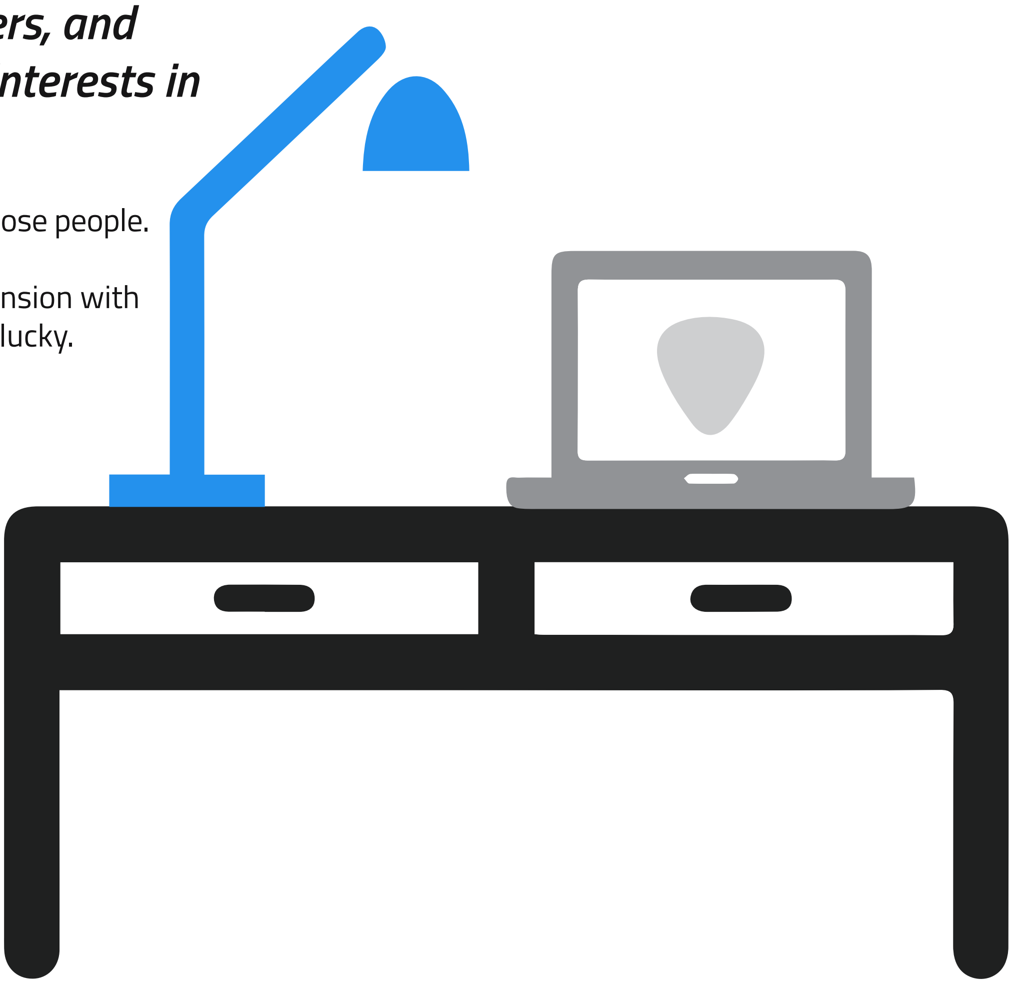
There's no right place to start with DevOps, but there are reasons that different people choose to start. Being aware of the people you are talking to and the processes they use can make your DevOps experiments more likely to grow into a business-wide success.

Not everyone likes DevOps for the same reasons. You probably know why DevOps sounds useful to you, but getting buy-in from other teams is tricky—and absolutely crucial. Knowing what others are likely to care about can make it easier to get them on board and make DevOps a reality.

***Maybe you already knew that Developers, Implementers, and Leaders have different interests in DevOps.***

After all, you probably are one of those people.

And if you've never experienced tension with another department, you are very lucky.



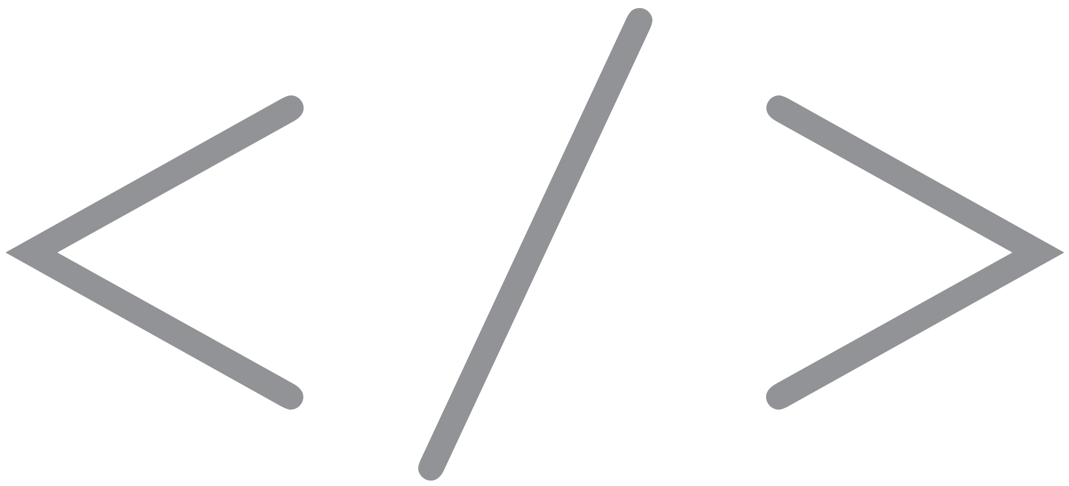
# </Developers>

Developers are most interested in optimizing resources, both in terms of people and infrastructure. This is because they do project- or sprint-based work where a schedule of milestones formalizes how they will be evaluated.

Unfortunately, developers have thus far seen less benefit from DevOps than Implementers or Leadership, perhaps because their involvement in deployment appears to be extra work that jeopardizes hitting milestones.

## If you are talking to a Developer:

Tell them about a development cycle where stable deployments mean less time fixing bugs. Better accounting for deployment work means they can give more accurate scoping and have a more consistent workflow.



## _Implementers

Not surprisingly, Implementers are disproportionately interested in automating release cycles and improving mean time to recovery. Whereas developer work is typically evaluated through a formal project management strategy, implementer work is commonly viewed through the binary lens of "is the site up or down." Implementers see the value of DevOps in preventing hiccups.

### If you are talking to an Implementer:

For years, implementers have been catching the pig after development tosses it over the fence. DevOps is the way to make that handoff more graceful. Tell them about a release process with greater visibility and faster recovery when something goes wrong.

| Source: Technet | IT Leaders | IT Implementers | IT Developers |
|---|---|---|---|
| Making Progress with DevOps | High | Med | Low |
| Have a DevOps Strategy | Low | Low-Med | Low |
| DevOps Will Optimize Resources | Med | Low-Med | High |
| DevOps Will Improve Quality | High | High | Med-High |
| DevOps Will Yield Faster Resources | Low-Med | Med | Med |
| DevOps Will Yield Faster Responses to Requirements | Med | High | High |

## *Leadership

IT Leadership may have higher expectations than Developers or Implementers for their companies' adoption of DevOps.

They may also be inclined to believe they are further along in the adoption process than they really are.

The disconnect between the perception and expectation of Leadership and the rest of the IT team is a potential source of resistance to DevOps—when one group thinks they are already halfway there while another thinks they have barely begun, the distance to the finish line appears drastically different.

IT Leadership is more interested in scaling operations and improving communication and collaboration than the other groups.

This is likely because they are more concerned with long term strategy and processes than people doing the day to day work on the next release.

### If you are talking to an IT Leader:

Talk about the long term benefits of investing in DevOps. Cultural change in particular is slow to come, but it also yields the biggest dividends.

### If you are talking to a random person:

Tell them that DevOps turns business requirements into feature releases sooner.

While the relative interest in a given benefit of DevOps varies between groups, the overall trends are fairly similar. Everyone is most interested in improved application/IT quality and more quickly responding to business requirements.

## Size Matters

Another variable in the DevOps adoption cycle is the size of the company. Given that DevOps is a new phenomenon, and that new processes and technologies are easiest to implement when a company is still being built, we might expect to find DevOps interest and success firmly rooted in startups and small to medium-sized businesses.

The focus on new technologies in DevOps blogs and talks—ones that are difficult to implement in legacy environments — might also give the impression that DevOps is mostly for new companies that have a high degree of freedom.

Surprisingly, the level of DevOps interest, adoption, and success from large business and enterprises is high. The interest in DevOps makes sense; in fact, DevOps makes the most sense for enterprises, who have the most complicated environments and the highest number of points of failure in moving from development to production.

The synchronization of Development and Operations, which can be taken for granted by small companies, addresses inefficiencies that grow with scale.
In a study conducted by Saugatuck Technology and sponsored by Microsoft and ScriptRock, companies with 1,000-4,999 employees reported significantly greater success adopting DevOps People, Processes, and Culture than those with <1,000 employees or >5,000.

Overall success was similar across all segments: approximately 20% of all polled said they have been "Very Successful" or "Extremely Successful" at adopting DevOps principles.

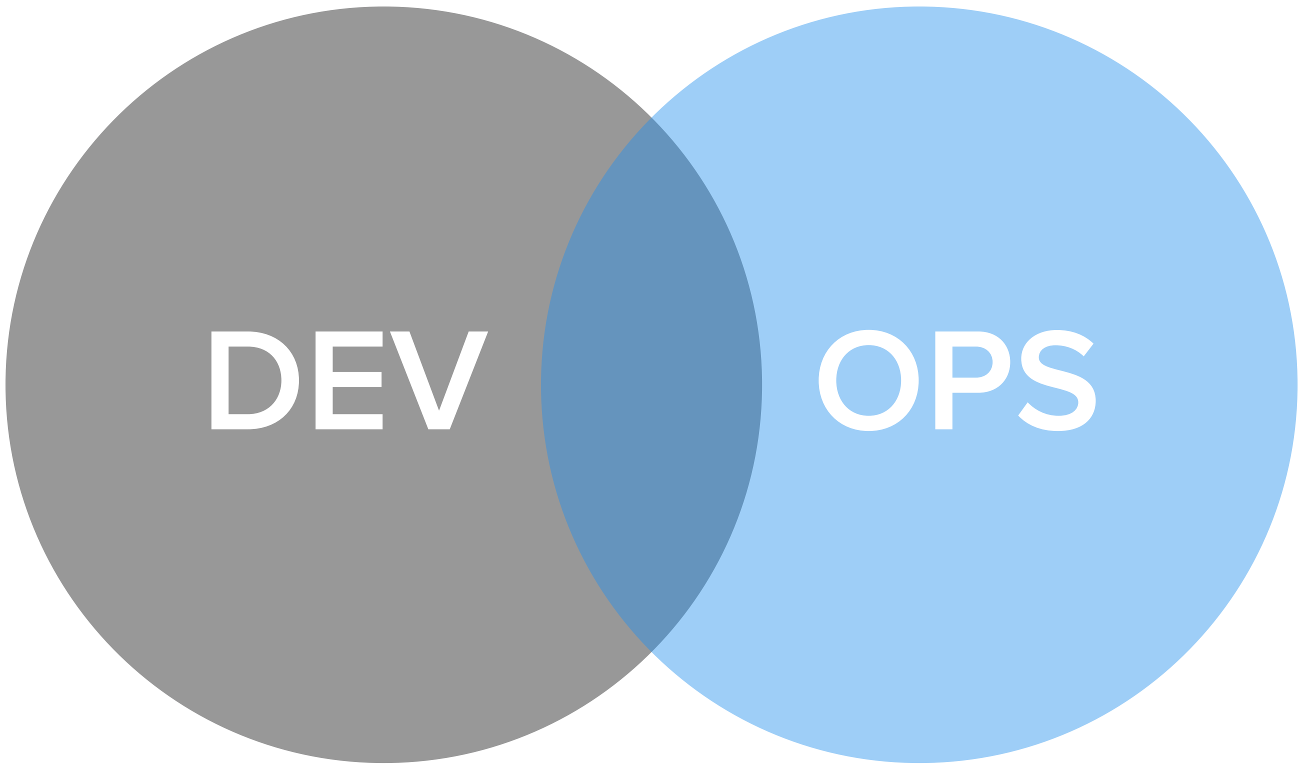
But while the smaller companies had success rates of 15%, 20%, and 18% with People, Process, and Culture, respectively, those same numbers were 20%, 23%, and 24% for large businesses.

## Why has DevOps been largely successful in large companies?

On one hand, the bureaucracy that emerges at scale can make it hard to introduce new ideas.

On the other, mature processes makes it easier to propagate processes once they have been approved.

While technological solutions are useful and faster to get running than cultural changes, the investments larger companies are making now will pay the largest dividends in the long run.



## The Downside

- Automation may seem like a miracle, but nothing is perfect.
- Automation tools absolutely, unrepentantly SUCK at collaboration.

### They are too hard to use.

Automation tools have notoriously steep learning curves. This restricts the number of users for that tool within your business, usually to a single team. Often to a single person. You cannot expect your devs and ops teams to play nice if one group holds the keys to your automations. It simply will not happen.

### They are a lousy medium for sharing information.

Recipes, manifests and YAML files are effective means of automating systems. But they are an awful way of storing desired state. Desired state should be visible. It should be easy to understand and simple to modify by any stakeholder.
If the systems that define your desired state are not open and accessible, then collaboration will suffer. When knowledge is locked up, collaboration breaks down. the keys to your automations. It simply will not happen.

### They inhibit collective innovation.

Enterprises embarking on an IT automation initiative are prone to replacing business improvement metrics with vanity metrics. Agility, speed to market and costs are harder to measure than “Number of applications automated.”
The automation of a system is often considered an “end,” not a “means.” Bad process and bad design become codified because they get buried. Taking the necessary steps to ensure collaboration on requirements is often pushed aside in the interests of simply getting it done.

***The dilemma of the enterprise is the dilemma of every software developer writ large. Either everyone understands what everyone else has done—both in engineering and operations—or something breaks and you have to untangle it in the way that is most costly to the business and most stressful for your team.***

***Automation done right provides enormous value for businesses. It goes without saying that automation in the enterprise is critical to keeping up with today’s dynamic business demands.***

***But there is a tendency for organizations to envision automation as the panacea for their DevOps aspirations, which can be a great kick in the ass for some large, entrenched Enterprises. At the same time, it can also lead to new tool silos emerging, skill gaps between key resources, and a potentially slower pace to market.***

## Caution Advised

Automation that kills collaboration equates to zero sum DevOps. It represents such a huge risk to the Enterprise because its effects can take months to become visible.

Be aware of the risks going in. Watch out for the warning signs. Don't sacrifice long term success for short term glory.

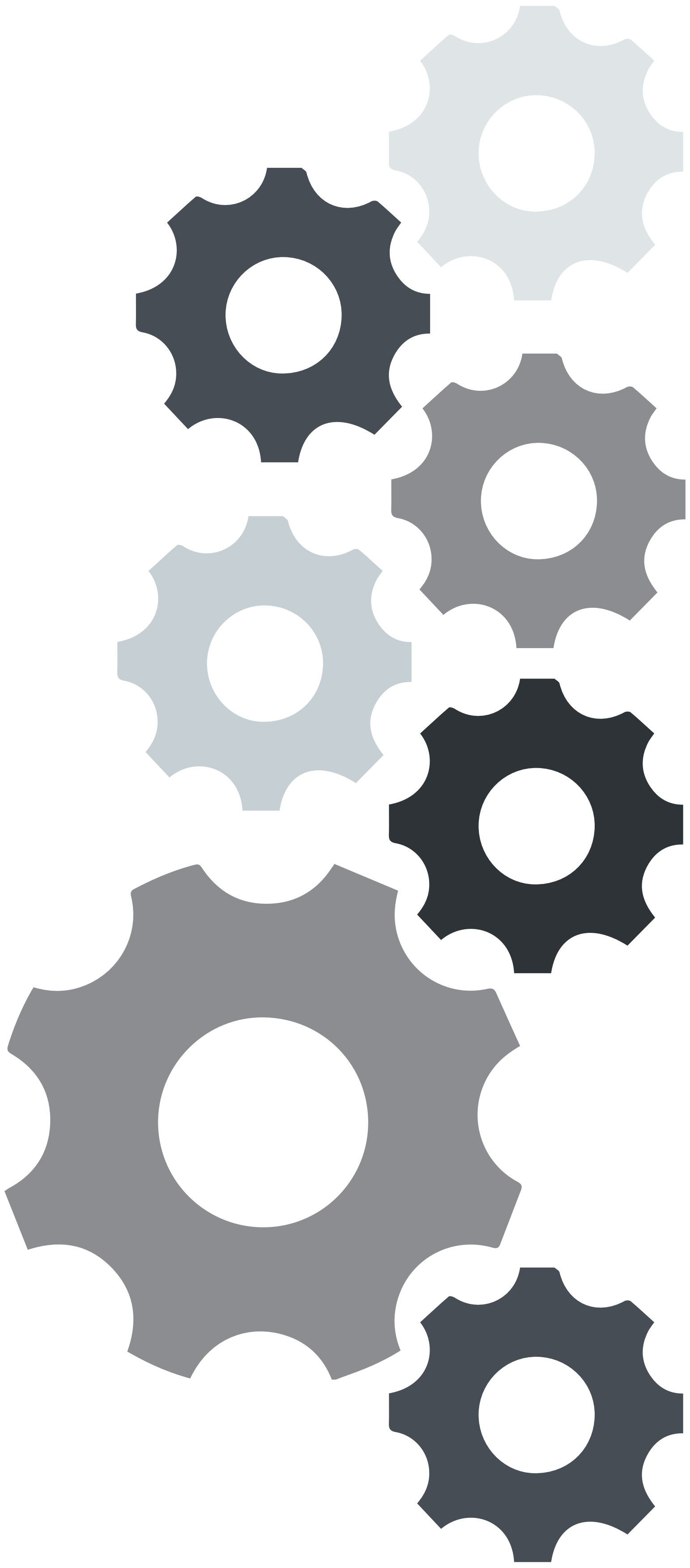
By 2016, according to Gartner, 75% of large enterprises will have more than four diverse automation technologies within their IT management portfolio, comprising any one of a number of solutions tailored specifically for the automation of one and/or multiple process types.

Enterprise job scheduling and workload automation solutions continue to represent the largest percentage of the IT automation market. Yet for most IT organizations, scripting remains the de facto standard of automation and the starting point for adopting an architectural approach to IT automation.

Wherever you have a scripted process, make sure the team knows what script to use for which purpose and how to update the scripts when the environment changes. The only thing worse than not automating is trying to automate too much.

The end result is that IT automation becomes a strategic priority that drives alignment and collaboration between organizational islands within the IT department and enables the marriage of revenue and productivity by reducing IT operational costs while improving staff productivity. All of these challenges must be dealt with in order to do automation and DevOps across your enterprise.

The main risk for most enterprises today is rushing full steam into automation without first laying the groundwork for your success.



## Visibility

If you cannot validate your state then you cannot gauge the success of your automation efforts or maintain their quality over time.

You are only truly ready for automation if you can also validate the key components of your configuration at any point in time.

To gain accountability for your apps and infrastructure configuration you need to first define this desired state.

***Don't automate what you don't understand.***

## Accountability

Preparation is more than choosing a tool and training your staff.

The real work is gathering requirements. With legacy infrastructure in play, what matters most is getting visibility of your current state.

It is tempting to jump straight to writing automation scripts but to do so without a full appreciation of your current state is a mistake.

***Don't automate what you cannot validate.***

## Automation

You are now ready for automation. The best part is that the solid foundation you laid down in the first two steps gives you the ultimate freedom in tool choice. Choose Puppet. Go with Chef. Try Ansible. Even Bash will work for now.

Your configuration is sound no matter how your builds are being done. The effects of a manual build will be just as visible, and just as verifiable, as those brought to life using the most elegant recipe, manifest or playbook.

Your automation efforts can be targeted and deliberate, and as such, they can remain under control.

*"Automation is necessary, but not sufficient, for Enterprises.*

*It doesn't buy you awareness."*

- Kevin Behr, co-author of "The Phoenix Project"

## Conclusion

DevOps is a simple ethos, but its wide appeal and increasingly fluid definitions can complicate deployment and integration. Although DevOps is more than just automation, the practicality and comprehensibility of automation tools make them a tempting stand-in for company-wide change, for businesses large and small.

Conceptually, it is problematic to view automation as an end and not a means; practically, this philosophy can not only hinder DevOps adoption, but can even completely disrupt it. The predictable pitfalls, including increased compartmentalization and accountability apathy, are numerous and costly.

It may seem tautological or even nonsensical, but a DevOps approach to DevOps is key. Preparation, open lines of communication, and an eye towards making automation work for everyone, from rank and file programmers to business leadership, are all crucial if you are going to succeed with DevOps. ■