Building a DevOps Culture

DevOps is as much about culture as it is about tools.

Mandi Walls

O'REILLY®

O'REILLY® Velocity Web Performance and Operations

Building a DevOps Culture

Mandi Walls

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

Building a DevOps Culture

by Mandi Walls

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

April 2013: First Edition

Revision History for the First Edition:

2013-04-15: First release

ISBN: 978-1-449-36417-5

[LSI]

Table of Contents

Introduction

After a few years of talking about "What is DevOps?", a general consensus has started to form around DevOps being a cultural movement combined with a number of software development practices that enable rapid development.

As a technical process, tools have emerged that enable teams to work more quickly and efficiently. Tools alone are not enough to create the collaborative environment that many of us refer to now as DevOps. However, creating such an environment is a huge challenge; it requires assessing and realigning how people think about their teams, the business, and the customers.

Putting together a new set of tools is simple when compared to changing organizational culture. DevOps is a departure from traditional software development practices for just this reason. By celebrating anti-social superstar developers, or allowing code to go to production that wasn't properly tested, or deciding that the only reason the code doesn't run is because Operations is stupid, we are not doing our best to enable the business and create value for our customers.

There are those among us who were "doing devops before DevOps." The most successful, rewarding projects I have worked on in my career were those that had an open, professional culture, but by attaching a common vocabulary, we can give others a blueprint for adjusting their organization's culture toward one that is more DevOps-like.

What Is Organizational Culture?

The concept of organizational culture was developed in the middle of the 20th century, ostensibly by Edgar Shein at MIT Sloan. The idea that a group of people working together in a corporate environment could create a culture distinct from the greater societal culture is now fairly well accepted in many industries. IBM and Walmart are well known for having cultural characteristics that separate them from competitors, for good or bad.

Groups of people create a culture through shared values and behaviors. How we reward behaviors, how we treat those values as malleable or immutable affects how strong the organization's culture is and how well it is supported by the participants. It also affects how the members of the culture will react to attempts to modify the characteristics of that culture.

The last few years have seen a lot of discussion of what DevOps is and isn't. DevOps is as much about culture as it is about tools, and culture is all about people. No two groups of people are guaranteed to create the same sort of culture under similar circumstances. So to talk about a cultural movement in absolute terms is disingenuous. Implementing a prescribed toolchain won't magically turn your team into a DevOps team. Using DevOps-friendly tools and workflows can help your team work in a more DevOps manner, but creating a culture that is supportive of the ideals of the DevOps movement is crucial.

We added a Velocity Culture track to the 2010 Velocity Conference out of recognition that there are important cultural aspects of successfully operating large infrastructures. What is more challenging, however, is how to help aspiring DevOps cultural warriors make modifications to their organizational cultures in order to reap some of the

benefits of DevOps. Cultural change in general is a topic widely discussed in business research, and we can borrow from lessons learned by other industries.

What Exactly Makes a Culture "DevOps"?

Culture, as an abstract collection of ideas and behaviors, is hard to pin down to a proscriptive set of requirements. In a technical team, it's much more easy to talk about what kinds of tools to use and how to use them; the tools themselves are accessible as constructs. The people that will use those tools are complex.

The traditional friction between teams thought of as "development" and teams thought of as "operations" boils down to a few key cultural characteristics.

Open Communication

A DevOps culture is one created through lots of discussion and debate. Traditionally siloed technical teams interact through complex ticketing systems and ritualistic request procedures, which may require director-level intervention. A team taking a more DevOps approach talks about the product throughout its lifecycle, discussing requirements, features, schedules, resources, and whatever else might come up. The focus is on the product, not building fiefdoms and amassing political power. Production and build metrics are available to everyone and prominently displayed. The current infrastructure is documented, and maybe someone went to the trouble of having it printed on a plotter, which is totally cool.

Incentive and Responsibility Alignment

One of the early controversial aspects of what became DevOps was the assertion that Engineering should be doing on-call rotations. In fact, this idea was presented in a way that made it sound like your developers would *want* to be on call if they were truly dedicated to building the best possible product, because they were the ones responsible for the code.

I don't remember now who first articulated this point; my personal DevOps epoch begins with John Allspaw and Paul Hammond's Flickr talk at Velocity 2009, and they may have mentioned it. Regardless, the cultural characteristic here is empowerment. The people who are empowered to directly make an improvement are the ones who are alerted to the defect.

Likewise, your team is incentivized around your core goal: creating an awesome product for your customers, whatever that product happens to be. Development isn't rewarded for writing lots of code. Operations isn't punished when all that code doesn't run as expected in production. The team is rewarded when the product is awesome, and shares in the improvement process when the product could be more awesome.

Respect

All team members should respect all the other team members. They don't actually all *have* to like each other, but everyone needs to recognize the contributions of everyone else, and treat their team members well. Respectful discussion and listening to other's opinions is a learning experience for everyone. No team member should be afraid to speak for fear of being abused or rudely dismissed.

Trust

Trust is a massive component of achieving a DevOps culture. Operations must trust that Development is doing what they are because it's the best plan for the success of the product. Development must trust that QA isn't really just there to sabotage their successes. The Product Manager trusts that Operations is going to give objective feedback and metrics after the next deployment. If any one part of the team doesn't trust another part of the team, your tools won't matter. Additionally,

if you don't trust the people who work for you, why are they working there? Why are you?

Ready, Set, Change

One thing is for sure: we cannot "greenfield" human beings the way we can tear down an application and rebuild it in another environment. We can't theorize about spherical cultures in a vacuum to simplify our equations. We see this every day in society; the residual effects of cultural change writ large in our day-to-day lives. The Civil Rights movement in the United States didn't wipe out the previous culture and start over; there are still people whose opinions about such things are "from another era."

People come with baggage: preconceived notions, past experiences, prejudices. When these individual characteristics are at odds with the culture that we want to foster in our organizations, we create stress. Alleviating that stress means either changing the person or changing the organization they belong to.

Changing the values of our organization is a challenging task, one that has been under study by management scholars for decades. There are numerous interesting case studies about disastrous cultural change initiatives, failed cultural integrations from mergers, and investigations into what makes cultures change. We are not alone in our struggles to better serve our businesses by improving our work cultures and processes. Searching on the Harvard Business Review website for "cultural change" will get you 60+ publications going back nearly 30 years.

One recent anecdote that I found particularly interesting was buried in a story from The Atlantic about recent trends in repatriating manufacturing work to the United States after years of moving those tasks abroad for labor cost savings.

The story of GE's GeoSpring water heater is fascinating, regardless of the underlying drama of globalization and outsourcing. The product was designed in the U.S. and the specifications were sent to Asia for production; the manufacturing process became a black box that the design team sent instructions to. When looking at how to manufacture this particular product in the U.S., they brought together "design engineers,…but also manufacturing engineers, line workers, staff from marketing and sales" to discuss how best to build it, and discovered the design was "terrible" and no one actually wanted to try to build it.

It sounds like Development wrote some code and "threw it over the wall" to Operations to figure out, doesn't it?

GE's integrated team sat down and redesigned the product with input from everyone, and ended up building a better product, at lower cost. It's what we want for our technology products too, the ability to best serve the business and create value. The cultural shift is acknowledging that everyone has valuable input and communicates. This was a DevOps cultural shift. It is a change in the value of some new behaviors versus the old ones.

Know Where You Are Starting

When we first join new organizations, we go through a learning process, discovering the mores and acceptable behaviors of that organization, be it a new company, a sports team, or the PTA. It's common to get different answers to questions like "Why are we doing X this way?" from different people. When we hope to change the prevailing culture of the group, we need to assess the current state of that culture, and ask all of the questions about why we're doing X the way we are.

The one question you must ask is "What is our primary business goal?" You need to know what people have decided is the motivating factor for the business as they know it. If the organization's leadership has done a good job of communicating the organization's goals, you'll find most people are on point. It's not uncommon, though, to hear some folks claim that the business goal is "We're a content company; we produce awesome content for our users" and others "We're an ads-placement company; we sell ads to make money for our advertisers." Or maybe, "We provide a stable application for our users" versus "We are driving innovation and leading the competition in new features." Are these goals mutually exclusive? No, but they represent a difference

in viewpoint that will color people's actions and decisions when it comes to the products they work on.

In an ideal situation, you'll have some sort of consensus. Even if you have two related goals, you're doing mostly OK. The comedy of errors starts when you have several divergent ideas of what the company is actually trying to accomplish. The reasons for this often include poor communication from management as to what the company is driving toward. It's also possible that fuzzy goals like "serving the customer in the best way possible" are different for different teams in your organization, especially if they have different "customers."

Now that you know the goal of your business, you can take this knowledge and focus your DevOps cultural change toward enabling this business and working toward this goal.

Know Why You Are Changing

Proving change is necessary requires some legwork. It's fine to want to change your organization because "everyone" is doing DevOps now, but you're looking at months of work, new tools to learn and implement, teams to restructure. These costs must be outweighed by the benefits, so you have to be able to put real value on your processes.

Articulating upfront what your goals are will help you with other phases of your DevOps roll out. Some common measurable goals are:

- Reduce time-to-market for new features.
- Increase overall availability of the product.
- Reduce the time it takes to deploy a software release.
- Increase the percentage of defects detected in testing before production release.
- Make more efficient use of hardware infrastructure.
- Provide performance and user feedback to the product manager in a more timely manner.

These goals have monetary impacts on your business. It costs you people-hours when your software releases take days. It costs you revenue when the product isn't available for use by your customers. It increases your operating costs to run your infrastructure in a non-efficient way. Taking what you know about these values, you can put a monetary value on improving any of them.

Additionally, all of these goals can have a numerical threshold attached to them. "Reduce deployment time from 10 hours to two hours." "Increase percentage of defects detected in testing from 25% to 50%." You can reason about these numbers and judge the success of your transition process.

These goals also give you a stable platform for obtaining support from the rest of your organization. Starting from metrics such as "It currently requires six team members 10 hours to deploy a new release of our product. This costs us $X for every release" provides a concrete number for comparison as you implement your DevOps changes. It attaches real value to a real pain point in a way that is more constructive than "Our deployments suck."

Executive Sponsorship

Getting most folks on the same page is a job best sent up the org chart as far as you can push it. In order to make changes to things like compensation and incentives, you need to get management buy-in at a high enough level that any necessary changes can be facilitated without a battle. This is one distinct advantage small organizations have over large ones, the ability to make tweaks to core human resources functions in order to influence the behavior of individuals.

Having an executive sponsor is also helpful when spreading the good news about your cultural adjustments. It gives your teams a feeling that they aren't going off on a hunt for DevOps nirvana only to have the plans changed on them a few months down the road. Resist the urge to start a grassroots movement; in this case, getting permission comes with more resources than asking forgiveness later.

Your executive sponsor is going to be particularly interested in the measurable goals you've developed, and will likely have their own that might fit in with your DevOps goals.

Technical Leadership

In the process of implementing a DevOps culture, you will likely be asking your technical staff to change their tools, behaviors, and workflows. You will want someone on hand to be your go-to person to talk about DevOps to your teams. It might be you! And that is awesome! It could also be an architect or other senior person who is well liked

and well respected, or a few of these folks so you don't overburden a single person.

This person, or team of people, will serve an important role, a combination of evangelist, tools expert, process subject-matter expert (SME), and buddy. They're like your camp counselors. They'll teach a little bit, answer questions, reassure the reluctant, and bring the marshmallows at the end of a long day. These folks are hard to find, and often aren't who you think they are. The last person you want for a job like this is someone who's smart but a complete jerk everyone hates but who happens to know how to use the tools. Find the people who everyone wishes were on their team, borrow some of their time, and form a working group to help other teams.

Without these folks to shepherd the various factions of your organization, it's easy to lose a few sheep along the way. Your SMEs can be there to build interpersonal relationships and open communication paths with outside teams, cutting down on friction and politics.

Pilot Projects

At this point you know you're going DevOps. Your management is on board and ready to help as necessary. Your tiger team is only a couple people so far, but they're enthusiastic and well respected. Now what?

Now you need a pilot project. You have to find somewhere to start in your project portfolio. You are going to transform not just the product that the pilot team is building, but the processes, workflows, and team dynamics. As you've been talking to people in your company, getting feedback on things like goals and who's awesome, you've probably piqued the curiosity of more than a few folks. You might have multiple projects that want to try out your fancy new DevOps processes that you can choose from to start a pilot.

What makes a good pilot candidate? Lots of different project characteristics can make a good pilot. Most importantly, you don't want to start too big the first time out. Rolling out the DevOps machine for a key central service is going to be hugely frustrating while you're figuring out which tools work best for your environment and how to improve your communications and processes.

You also don't want something that is too small. It's easy to dismiss a tiny pilot project with "Oh, sure, that team can do DevOps because

they've only got five people. We have six in QA alone; it will never work!"

The first time out, try for a project team that is collocated. Changing up process and communication is hard; doing it across five timezones is a good hurdle to avoid if you possibly can. If all your teams are distributed, look for teams with a long history of working successfully across long distances. Their team culture may already reflect some of the characteristics of a DevOps culture that you're looking for.

Set Goals

Remember those goals from earlier? Now you're going to apply them directly to your pilot project.

Choose the goals that most make sense for the individual project, be it reducing time to market, increasing efficiency, reducing deployment time, or whatever. Talk with the project team. They have problems they want to improve, pain that needs to be alleviated.

Finally, give these goals a timeline. If your organization is using an agile methodology, work with that, do sprints, reassess, set new goals, keep improving incrementally. You'll be taking time to install and learn new tools and processes, so at the beginning work may seem slower than you think it should be.

Mentoring the New Culture

As your pilot project progresses, you'll be mentoring the product team in the values that foster a DevOps environment: communication, responsibility, respect, and trust.

Fostering open communication channels is the foundation of a successful DevOps culture. One of the most common complaints from many Operations teams is that they weren't aware of an impending change, or weren't told about a new product feature. When I have asked these folks if they attend the Development team's standup meetings, the answer I typically hear is "No, we don't go to their meetings, they never talk about anything that concerns Operations." Open communication leads to better collaboration throughout the entire team. Every team member should know that he or she can go to any other member of the team to ask a question, and expect to get an answer or some kind of help in finding one.

Your new DevOps-style standup meeting includes everyone. Your developers, your QA engineers, your system administrators, all of them get together to discuss what they're working on. The benefit of this single change is immense, and even if you're reading this article and thinking "This isn't for me," bringing a full team together to intentionally talk about the product on a regular basis is going to help you build a better product.

Maybe your developers are working on how the product is going to use a new third-party API. They're talking about credentials and authorization with that third party. QA will need to know how to write tests to flex the features; do they need fake accounts or dummy users? Your sysadmins will have questions about the SLA of the API, how failures will be monitored and rolled up for reporting. What if the API

has an outage? The whole team has worked on the metrics and information necessary for the product manager to make an informed decision on disabling the feature. When everyone knows up front what new features are in the pipeline, it will be less likely for any of these topics to be missed. This makes your product more reliable and gives your customers a better experience.

Another aspect of opening communications is logistical. Your best bet for successful communications is with people who sit together or have a common physical space that allows for chance interactions and conversations. Think about the conferences you have attended. The "hallway track" is often just as awesome as the talks. People get to meet in person and talk in real time about interesting things. The product your team is building benefits from this unfettered communications. Get your folks some dedicated white boards to work on and share ideas. Facilitate all avenues of communication.

If your team is not permanently collocated, bring them together for as long as you can at the beginning of the project. You want them to get to know each other, talk to each other, and begin to gel as a team. A week is probably the minimum; two is better.

Your next hurdle is realigning the responsibilities of your team. The scariest part of this step is fixing your broken on-call schedule. Guaranteed, if only your system administrators have ever been paged about your product, your responsibility model needs to be updated. It should be obvious; if the desire is to have the product be as awesome as it possibly can be, every bump, burp, or error in the code would report back to the person who wrote that code and is most likely to understand what is wrong in order to start the process of fixing it. Fault recovery processes in siloed teams are full of wasted time while additional input is requested from people who didn't think they needed to be available.

Similarly, Dev and QA shouldn't be the only ones creating tests for the product. If your team is adopting DevOps practices in order to put more product features in front of customers more often, you want to be confident that the way you're installing the product's requirements or new infrastructure will behave properly with your product. This requires testing. How you test varies with the availability of hardware resources and the tools your team might be using. Making it a priority to test infrastructure changes is assigning value to a new behavior, changing the incumbent culture.

Mentoring your team around respect derives from how you resolve interpersonal conflicts among team members. Team members need to feel that everyone is being treated fairly and has a voice. Sometimes teams fight, and fighting it out over a difficult topic isn't always a bad thing. But it's possible to have a good fight and a bad fight. Personal attacks and name-calling are obviously not the kinds of behaviors you want in a collaborative team, and you have to deal with them from a managerial standpoint. These are the difficult discussions for many managers, and it is tempting to sit back and let people deal with their conflicts. Do not be tempted to ignore destructive behaviors; engage the people who are fighting in a constructive discussion, work through the sticking points, and mediate a compromise if one is warranted. This is a good place to ask one of your SMEs to help mediate and act as an impartial judge.

Finally, your team members will learn to trust each other over time. The changes to behaviors in communication, responsibility, and respect build an open, collaborative environment. Your team will build new relationships that they wouldn't have had in an otherwise siloed environment. It may take a while for your team to heal from whatever post-deploy PTSD may be haunting their nightmares.

Checkpoints and Declaring Victory

Your new DevOps-enabled team is going to be busy. They're learning how to use some new tools, they are learning to work together in more productive and efficient ways. You want them to know when things are going well.

Part of choosing good goals, with metrics attached to them, is that you can put up some graphs and see how the team is improving toward those goals. You can prove to the team that deployment time has improved, and by how much. The product manager is hopefully ecstatic that the new features customers have been suggesting are now in sight for making it into production on a more efficient timeline.

In addition to the metrics related to your goals, you also need to check in with your team members, one-on-one. While we all hope that the team is excited to be working more closely and getting a better product out, you still have to check on everyone and see how they are doing as individual people.

Finally, as a manager, you get to declare victory and throw a party. After working through a multifaceted project, it's important to stop, recap, and celebrate. Choose a milestone that represents a significant amount of work, and plan to pause. When facilitating a large cultural and behavioral change, you want a little bit of time to assess the impacts of these changes, not just to the team directly involved, but also to any tangential teams. Hopefully other teams are getting interested in changing their behaviors, and they can come to you, your team, and your SMEs for help. Creating some documentation around lessons learned and the challenges you faced will help other teams heading in a DevOps direction.

Then buy your team some cupcakes, or go to the movies, or do *anything* that says "We just did something difficult. We learned good new things, we unlearned bad old things, and we created a great product. Together."

include::colo.asciidoc

About the Author

Mandi Walls is a Senior Consultant with Opscode. Prior to joining Opscode in 2011, Mandi administered large web properties for AOL, including AOL.com, Moviefone.com, and games.com. She has a MS in Computer Science from The George Washington University and an MBA from UNC-Kenan Flagler.