DevOps Lessons for CIOS

UpGuard

Table of Contents

I. The State of DevOps

There is no doubt that the DevOps movement has gone mainstream. When even IBM and HP are dedicating sites to it there is no longer any question. If we were to place it on the Gartner Hype Cycle even the most devoted proponents would have to admit that it's rapidly approaching the "Peak of Inflated Expectations". What does this mean for you as a CIO? Should you steer clear of the movement entirely until things calm down a bit? Not at all. Should you be cautious in your approach to "implementing" DevOps though? Absolutely.

With vendors piling into an area that remains poorly defined the risk of making a misstep on strategy is high. It cannot be ignored though, the rewards in IT efficiency and quality improvements of well implemented DevOps initiatives are high. In a world of diminishing margins any opportunity for competitive advantage must be explored. Nascent as the movement may still be, especially for the Enterprise, some clear lessons have already emerged. Understanding these before beginning may be the difference between success and failure for your DevOps initiative and, ultimately, your entire IT strategy.

This eBook will help you avoid the most common mistakes.

Understanding these before beginning may be the difference between success and failure for your DevOps initiative and, ultimately, your entire IT strategy.

II. Define Devops Correctly

Much has been written on this topic but its importance means repetition is required. DevOps is not a product. It is not a job title. It is not a process. It is perhaps best understood by understanding the forces that brought it to life. In particular the increasing disconnect between the development and operations functions within IT departments. Despite ultimately playing for the same team, their individual goals of improving product through change (development) and maintaining service quality and uptime (operations), mean that they are frequently at odds. In most companies a “wall” has formed between them and no one, least of all the company as a whole, benefits from the battles that rage as a result.

First and foremost then, DevOps is any effort within an organization to align the goals of developers and operations staff within an organization. In practice this can be cultural or process-driven. It can be enabled by, but should not be centered on, the implementation of certain tools, particularly automation tools.

III. Collaboration First, Especially for Tools

Changing culture is hard. Buying tools is easy, and with the big boys now weighing in on DevOps the old saying, "No one ever got fired for buying IBM/ Microsoft..." is once again in play. Tools play a very important role in any DevOps initiative but they must complement, if not follow, cultural changes within an organization. Many a startup will point to their use of Puppet or Jenkins when asked about how they do DevOps. This won't cut it in the Enterprise, and may even be harmful (see "Tools as White Elephants" later in this eBook).

In its simplest form you can break down DevOps into two main elements, collaboration and automation*. It's no accident that collaboration is listed first here. Starting with automation can make matters worse. This is perhaps best illustrated with an example I witnessed first hand. An application manager within a company I once consulted with decided he would take up the DevOps torch for his organisation by starting to use Chef, a developer friendly configuration automation tool, for the project he was working on. After his team overcame the not insignificant hurdle of Chef's steep learning curve he was very satisfied with the overall improvement he saw in the build process within his development and test environments.

The problem came from the fact that there was no consultation with operations on this change. Unsurprisingly, the operations team decided that they wanted nothing to do with Chef and that they would not accept Chef builds into their Staging or Production environments. The benefits shown were rendered moot and use of Chef was discontinued.

* At UpGuard we believe you should add "Validation" in between the two. Don't automate what you haven't tested!

To avoid similar mistakes remember this:

- An initiative is not DevOps if it is wholly contained within one group.
- It is OK to lead with the implementation of a tool if agreement is reached on both sides that it is the right tool for the job.
- Avoid tools that enforce silos at all costs. A cross-functional tool that is only used by one team just makes matters worse.

IV. Don't Create Another Team

This one shouldn't really need mentioning, unless you're a 25 year old CIO wunderkind you would have seen this before. Take Service Oriented Architecture (SOA) and Business Process Management (BPM). Both worthy initiatives in and of themselves, but both acronyms that fell victim to unnecessary hype and broken promises from eager vendors. Tooling errors aside one key lesson learned from the companies that fell deepest in each of these areas was that building their initiatives around new, separate teams had at best negligent effect. At worst it made success impossible. Why did they think it would work? Well a central team would guarantee focus. It would consolidate and emphasize the requisite skills. A more cynical analysis would be that it represented an easy KPI tick for the executive responsible.

The reasons it didn't work are the same reasons it won't work for your DevOps initiative. By removing the function from your main development and operations team you remove their commitment to it. It becomes someone else's problem. Worse still, they know that if it works it is someone else's success. There is no need to point out what this means for collaboration between those teams and their new DevOps colleagues.

All you've achieved is the creation of a new silo. A silo with noble goals but one that is doomed from the start. Again, DevOps is first and foremost about collaboration. In an environment where collaboration between teams is sub par introducing a new team will only make matters worse. DevOps must be everyone's responsibility, everyone's initiative and, ultimately, everyone's success.

V. It’s Not Just DevOps

What’s in a word?

The simplicity and directness of the term DevOps is a beautiful thing. You’d be hard pressed coming up with a better word to describe the goal of aligning the interests and work of your development and operations teams. There is a problem here though. As an Enterprise CIO your IT team is much more than development and operations. There is little doubt that these two groups stand to gain the most from a little relationship counseling but answer this question. Do your developers have nothing but kind words to say about your QA team? Do your operations staff regularly sing the praises of security? Are your architects loved by anyone but themselves? Dismantling the wall between development and operations is a great place to start when dealing with a dysfunctional IT department. Walls exist between other teams as well though. Walls that not only won’t be addressed by a narrowly focussed DevOps initiative, they will likely be strengthened. Can improved collaboration and automation benefit other areas? Of course it can. Don’t make DevOps another silo. Include other teams wherever possible. You should even consider using another term to define any work in this space.

All you’ve achieved is the creation of a new silo. A silo with noble goals but one that is doomed from the start. Again, DevOps is first and foremost about collaboration. In an environment where collaboration between teams is sub par introducing a new team will only make matters worse. DevOps must be everyone’s responsibility, everyone’s initiative and, ultimately, everyone’s success.

VI. Don’t Throw Away ITIL

There has been some noise around DevOps representing the death of ITIL. As was the case with Mark Twain, these rumors have been greatly exaggerated. The truth is that the DevOps movement was not born in the Enterprise. It’s early proponents are unlikely to have much knowledge of, let alone experience with, ITIL. As an Enterprise CIO you will have seen the positive effect an ITIL implementation can have on an organization. When implemented correctly, and pragmatically, ITIL brings order where chaos reigned. DevOps devotees may wince at the perceived constraints but we know that the processes now form an essential part in most organizations.

The detail behind implementing DevOps within an ITIL organization can be found in a previous eBook of ours called, the *ITIL Guide to DevOps*. It will suffice here to point out that the practical way to view DevOps is as a means of process improvement for ITIL. Over time, improved collaboration and automation will reduce your reliance on ITIL processes. At Enterprise scale though they will not replace them. Most large companies who succeed with DevOps will do so on a foundation of ITIL.