# The ITIL Guide to DevOps

UpGuard

# Table of Contents

# Introduction

You're never safe in Enterprise IT. Just when you feel you've gotten a handle on the last hot topic you're hit with another. SOA, BPM, Agile, ITIL. You feel like screaming "Enough!" but you know resistance is futile. Gartner have said it's important so you know full well that you'll be asked to "do" it by management. The latest buzzword in IT is DevOps. DevOps is more philosophy than process. A recognition that the relationship between developers and operations staff was broken. Developers are expected to ship more code, more frequently. Change is their friend. Ops are expected to keep everything running smoothly, to protect uptime. Change is their enemy. Something had to give.

At its core DevOps is simply about improving collaboration between development and operations teams. As an extension of Agile it means including system administrators in the agile development process. In practice DevOps has also come to be associated with various tools as well. In particular infrastructure automation, automated testing, CI/CD and monitoring tools. You're in the Enterprise though. Where do you start with DevOps? As most Enterprises these days run on ITIL processes this is a pretty good place to start. Despite the rumours, DevOps doesn't mean the end of ITIL. DevOps should be seen as a means of ITIL process improvement. By using ITIL processes as a basis for a DevOps initiative you can greatly simplify and focus your efforts.

This eBook will step through the major ITIL processes, outline what typically goes wrong, and examine how application of DevOps principles and tools can help.

***DevOps is more philosophy than process.***

# II. Change Management

## Objective

The primary objective of Change Management is to enable beneficial changes to be made, with minimum disruption to IT services.

## What Goes Wrong?

What doesn't go wrong with Change Management in a typical Enterprise? It's the process that represents the very issue that gave birth to the DevOps movement, the clash between improving systems and keeping them running. Stakeholders are involved too late. Everything revolves around the CAB (Change Advisory Board) and, while it should be more forward looking it usually defaults back to a weekly change approval process where stakeholders protect their turf by blocking changes. The change process is a necessary evil but it too often becomes the major bottleneck holding back progress within an IT department. Errors due to change are also a major issue. Validations are often done manually and regression testing is frequently insufficient.

## How Can DevOps Help?

If your change process is a bottleneck then chance are it's because that's all it is to your Enterprise, a process. You've followed all the advice of ITIL and have appropriate prioritizations, approvals and stakeholders but guess what? You've taken the human element out. Email approvals are handy but people are much more likely to reject approvals when they haven't been consulted appropriately and are not face to face with the person responsible.

So the best "DevOps" initiative we can implement within change may not really feel like DevOps at all. It's simply good practice to improve collaboration between stakeholders in the process. The main goal is to make consultations between groups proactive, not reactive. One of the best examples of this I've seen was painfully simple. The CIO mandated that all CABs be face to face (where possible), with webcams used when not. Another involved requiring pre-approval from stakeholders for all changes with "reject" removed as an option. Stakeholders could instead request "More Info", in which case it was up to the change owner to consult with them to get their OK to proceed. This simple modification to the process resulted in a 60% drop in changes being rejected prior to release.

## Standard Change

As a company's DevOps practice progresses, the introduction of Continuous Integration, and eventually Continuous Delivery, should have the ultimate goal of pushing as much change into the Standard Change bucket as possible. ie: If it integrates and all tests pass then push it.

## Tools

We will cover in upcoming sections various automation tools that, whilst not directly affecting change, will have beneficial flow on effects. One key question to ask yourself regarding tools and change though is this, "Can any of the approval points in my Change Management process be automated?"

# III. Release and Deploy Management

## Objective

To plan, schedule and control the movement of releases to test and live environments.

## What Goes Wrong?

It's important to note that the objective lists both live and test environments. At the end of the day you want this process to be as smooth as possible. It rarely is, and the hidden killer in most Enterprises is the fact that a lot of this wasted time takes place when migrating between non-Production environments. Who hasn't gone through hours of troubleshooting to work out why the application that worked fine in UAT doesn't work in Staging? How many PMs have sat on the sidelines in frustration as their project gets held up while the team scrambles to get a QA or performance environment ready? An environment that is supposed to be always available. Another major issue comes from the management of conflicting projects. Schedules are altered and additional time factored in to mitigate poorly defined and implemented integration processes.

## How Can DevOps Help?

This is an area where the automation side of DevOps really comes into play. Automated deployment, automated testing and continuous integration are key. When managing environments and moving applications through them consistency is what you want. The first step towards that is consistent builds. Automated deployment tools are a must. No PM wants to hear that it will take 3 days to get UAT ready for testing. Tell them it will be deployed, validated and ready to go in a matter of minutes and you'll be assured of a seat at the project completion lunch.

How about keeping it that way? Automated testing, both functional and configuration will help maintain consistency (or prevent promotion of issues in the case of functional testing). One sign of an advanced IT shop is the implementation of monitoring solutions earlier and earlier in the process. Proactive identification of problems and inconsistencies is the key. Painful code merges and unecessary scheduling delays can be mitigated by the use of continuous integration tools, where all development branches are regularly merged into an integration branch and automated tests run to catch conflicts early and reduce regression testing requirements when it is time to go live.

## Tools

**Automation:** Puppet, Chef, CFEngine, Ansible, Salt Stack

**Functional Testing:** Cucumber, QTP, Rational, Selenium,

**Visual Studio Test Professional Configuration Testing:** UpGuard

**Continuous Integration:** Jenkins, Travis CI, Circle CI

# IV. Incident Management

## Objective
The primary objective of Incident Management is to return the IT service to users as quickly as possible.

## What Goes Wrong?
Any form of breakdown in communication between groups can translate into serious issues for the Incident Management process. Silos are often reinforced here as finger pointing and the "Blame Game" becomes common when what is required is a collaborative and unified effort to get systems back online and working correctly as quickly as possible.

## How Can DevOps Help?
Like the Change Management process, the Incident Management process will naturally receive flow on benefits in the form of reduced incident numbers when initiatives in other areas such as automated testing, deployment and continuous integration are put into action. Perhaps the best example of how a DevOps mindset can assist the Incident Management process though is the practice of making sure that development staff also go on call. The benefits from this are twofold. Firstly, seeing development staff playing their part and standing on the front line improves the camaraderie with the operations team. Secondly, an understanding of the impact their code has on supportability makes developers better coders as well.

## Tools
**Get the developer a pager :)**

# V. Knowledge Management

## Objective

The primary purpose of Knowledge Management is to improve efficiency by reducing the need to rediscover knowledge.

## What Goes Wrong?

This is an area that has long been a problem for Enterprise IT. Knowledge has typically been consigned to Word documents that no one reads, and which are out of date as soon as they are created. In most instances Knowledge Management is simply a case of making every effort to ensure that the best and brightest stay with the company, as the most valuable source of knowledge is locked in these employees' heads.

## How Can DevOps Help?

Automation is the key here. Whether you are dealing with a deployment, a test, or an approval, each time you automate something you are, effectively, capturing knowledge. ITIL may disagree with this definition but it is the only practical one. Executable Documentation creates a virtuous cycle; the documentation tests the system, and the system tests the documentation.

## Tools

The same tools we listed above for automation (deployment and test) are relevant here. Whilst biased I will say that the majority of them are lacking in one key area though: Collaboration. Capturing knowledge in a tool like Puppet or Chef is a good thing. If you can't use these tools though, you cannot view or contribute to it. Practical Knowledge Management is the intersection of both automation and collaboration. Tools like UpGuard that abstract away technical detail and actively promote collaboration are the future of this space.

# VI. Conclusion

IT fads come and go, and you should always be wary of the latest buzzword. Whilst time will tell whether the term DevOps will endure, its lessons on collaboration and automation certainly will. Adoption of DevOps practices within the Enterprise does not mean throwing out a perfectly good framework like ITIL. Think of DevOps as more of an evolution. ITIL brought order where there was once chaos. In some cases its formality and rigidity has gone too far though.

Application of Agile and DevOps principles over the foundation of ITIL can overcome this. By using your ITIL processes as starting points for analysis of potential benefits, an Enterprise DevOps implementation is given greater focus. Identify pain points that could benefit from greater collaboration and automation. Prioritize them based on potential benefit and estimated effort. Get started today though, otherwise your business will most certainly be left behind. ■