DevOps from a Sysadmin Perspective

PATRICK DEBOIS

Patrick Debois is a senior independent consultant, who has made a habit of changing both his consultancy role and the domain in which he works: sometimes as a developer, manager, sysadmin, tester, or even as customer. The one thing that annoys him most is the great divide between all these groups. Mr. Debois first presented concepts on agile infrastructure at Agile 2008 in Toronto, and in 2009 he organized the first DevOpsdays conference. Since then he has been promoting the notion of DevOps to exchange ideas between these different organizational groups and show how they can help each other achieve better results in business. He can be found via his blog, http://jedi.be/blog, and twitter, @patrickdebois. Patrick.Debois@jedi.be

While there is not one true definition of DevOps (similar to cloud) [0], four of its key points resolve around culture, automation, measurement, and sharing (CAMS) [1]. In this article, I will show how this affects the traditional thinking of the sysadmin.

As a sysadmin you are probably familiar with the automation and measurement part: it has been good and professional practice to script/automate work to make things faster and repeatable, and gathering metrics and doing monitoring is an integral part of the job to make sure things are running smoothly.

The Pain

For many years, operations, of which the sysadmin is usually part, has been seen as an endpoint in the software delivery process: developers code new functionality during a project in isolation from operations and, once the software is considered finished, it is presented to the operations department to run it.

During deployment a lot of issues tend to surface. some typical examples are that the development and test environment are not representative of the production environment, or not enough thought has been given to backup and restore strategies. Often it is too late in the project to change much of the architecture and structure of the code and it gives way to many fixes and ad hoc solutions. This friction has created disrespect between the two groups: developers feel that operations knows nothing about software, and operations feels that developers know nothing about running servers. Management tends to isolate the two groups from each other, keeping the interaction to the minimum required. The result is a “wall of confusion” [2].

Culture of Collaboration

Historically, two drivers have propelled DevOps: Agile Development, which led in many companies to many more deployments than operations was used to, and cloud and large-scale Web operations, where the scale required a much closer collaboration between development and operations.

When things really go wrong, organizations often create a multi-disciplined task force to tackle production problems. The truth is that in today’s IT, environments have become so complex that they can’t be understood by one person or even one group. Therefore, instead of separating developers and operations as we used to do, we need to bring them together more closely; we need more practice, and our motto should be, “If it’s hard, do it more often.”

DevOps recognizes that software only provides value if it's running in production. And running a server without software does not provide value either. Development and operations are both working to serve the customer, not to run their own departments.

Although many sysadmins have been collaborating with other departments, this has never been seen as a strategic advantage. The cultural part of DevOps seeks to promote this constant collaboration across silos, in order to better meet business demands. It goes for "friction-less" IT and promotes the cross-departmental/cross-disciplinary approach.

A good place to get started with collaboration is places where the discussion often escalates: deployment, packaging, testing, monitoring, and building environments. These places can be seen as boundary objects [3]: places about which every silo has its own understanding. These are exactly the places where technical debt accumulates, so they should contain real pain issues.

Culture of Sharing

Silos exist in many forms in the organization, not only between developers and operations. In some organizations there are even silos inside operations: network, security, storage, servers groups avoid collaboration, and each works in its own world. This has been referred to as the Ops-Ops problem. So in geek-speak, DevOps is actually a wildcard for *dev*ops* collaboration.

DevOps doesn't mean all sysadmins need to know how to code software now, or all developers need to know how to install a server. But by collaborating constantly, both groups can learn from each other and rely on each other to do the work. A similar approach has been promoted by Agile Development between developers and testers. DevOps can be seen as the extension of bringing system administrators into the Agile equation.

Starting the conversation sometimes takes courage, but think about the benefits: you get to learn the application as it grows, and you can actively shape it by providing your input during the process. A sysadmin has a lot to offer to the developers: for instance, you have the knowledge of what production looks like, and therefore you can build a representative environment in test/dev. You can be involved in load testing, failover testing. Or you can set up a monitoring system that developers can use to see what's wrong. You can provide access to production logs so developers can understand real-world usage.

A great way to share information and knowledge is by pairing with developers or colleagues: while you are deploying code he comments on what the impact is on the code and this allows you to directly ask questions. This interaction is of great value in understanding both worlds better.

Revisiting Automation

As specified in the Agile Manifesto [4], DevOps values "individuals and interactions over processes and tools." The great thing about tools, as opposed to culture, is that they are concrete and can have a direct benefit. It was hard to grasp the impact of virtualization and the cloud unless you started doing it. Tools can shape the way we work and consequently change our behavior.

A good example is configuration management and infrastructure as code. A lot of people rave about its flexibility and power for automation. If you look beyond the effect of saving time, you will find that it also has great sharing aspects: it has created a "shared" language that allows you to exchange the way you manage systems with colleagues and even outside your company by publishing recipes/cookbooks on GitHub. Because we use concepts such as version control and testing, we have a common problem space with developers. And, most importantly, automation is freeing us from the trivial stuff and allows us to discuss and focus on what really matters [5].

Revisiting Metrics

Measuring the effects of collaboration can't be done by measuring the number of interactions; after all, more interaction doesn't mean a better party. It's similar to a black hole; you have to look at the objects nearby [6]. So how do you see that things are improving? You collect metrics about the number of incidents, failed deploys, number of successful deploys, number of tickets. Instead of keeping this information in your own silo, you radiate these stats to the other parts of the company so they can learn from them. Celebrate successes and learn from failures. Do post-mortems with all parties involved and find ways to improve. Again, this changes the focus of metrics and monitoring from making fast repairs to supplying feedback to the whole organization. Aim to optimize the whole instead of only your own part.

The Secret Sauce

Several of the "new" companies have been leaders in these practices. Amazon with their two-pizza team approach [7] and Flickr with their 10 deploys a day [8] were front-runners in the field, but more traditional companies such as National Instruments are also seeing the value of this culture of collaboration. They see collaboration as the "secret sauce" that will set them apart from their competition [9].

Why? Because it recognizes the individual not as a resource but as resourceful enough to tackle the challenges that exist in this complex world called IT.

References

[0] Patrick Debois: http://www.jedi.be/blog/2011/09/06/DevOpsdays-melbourne-keynote/.

[1] John Willis: http://www.opscode.com/blog/2010/07/16/what-DevOps-means-to-me/.

[2] Damon Edwards: http://dev2ops.org/blog/2010/2/22/what-is-DevOps.html.

[3] Israel Gat: http://theagileexecutive.com/2010/07/06/boundary-objects-in-DevOps/.

[4] http://agilemanifesto.org/.

[5] Ernest Mueller: http://blog.cutter.com/2011/09/11/originality-and-operations/.

[6] Patrick Debois: http://www.jedi.be/blog/2011/09/06/velocityconf-DevOps-metrics/.

[7] http://highscalability.com/amazon-architecture.

[8] John Allspaw and Paul Hammond: http://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr.

[9] Jesse Robbins: http://radar.oreilly.com/2007/10/operations-is-a-competitive-ad.html.