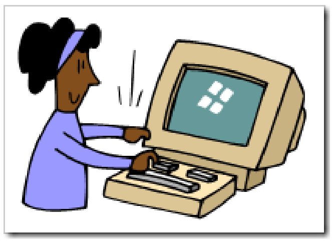
DevOps. It&rsquo;s in the Culture, Not Tech.

Don MacVittie, 2011-29-11



#F5 DevOps – Managers need to make use of existing technology and adopt culture change.

It is entertaining to read all that is currently being written about DevOps. Having been a developer, a development manager, an operations manager, and even a CTO, I can attest to the fact that the “throw it over the wall” syndrome is real, and causes real problems for everyone involved. That is about where my agreement with the current round of pundits ends. The thing is that they talk like there is some fundamental technological reason why DevOps isn’t happening. That’s just not true. For those a little behind in your jargon, DevOps is making operations prevalent in the decisions of your development organization.

We’ll take the discussion a little bit at a time. We have virtualization. We have astoundingly good virtualization from VMWare, Microsoft, RedHat, et al. So many of the concerns about development and ops go away immediately. “Developers test in the environment their tools run in!'” is oft-heard in the DevOps conversations. But that’s just not an issue. They can run three different OS’s to test with, and as many browsers in each OS as you want to support – all with a single set of hardware. So make testing in your operational environment mandatory. In fact, for most tools, they’re developing on whatever OS is being targeted anyway, because there are subtle differences – or no support at all – in other OS’s.

Virtualization taken hand-in-hand with the capabilities of an Application Delivery Controller (ADC) like F5’s BIG-IP can also remove some of the “throw it over the wall” symptoms easily – particularly in Agile or other high-rev environments. Take a copy of the VM running the app today, modify that copy, place it on the network, then, utilizing one of the many algorithms available for load-balancing, switch a select load to the new server. Make it 1/3rd as likely as any other server to receive a given connection, and utilize persistence so users aren’t bouncing back and forth between revisions of the app. See how it performs under real-life usage. Have the Devs there for the first half hour, then make a “deploy/don’t deploy” decision, and if not deploying, take down the copy with the new code on it so the devs can continue to work on it. If deploying, then bring up copies of the new server and bleed connections off of the old ones, then bring those virtuals down and archive or destroy them, as your organization sees fit.

Most of the issues with DevOps are gone at that point. Testing is the only other issue that comes up a lot, and let’s talk frankly about testing. There just aren’t enough really good test tools out there. A test tool needs to automate as close to 100% of the testing process as possible in order for thorough testing to be feasible. The complexity of today’s applications leave most test tools in the dust. When Microsoft had MS-Test available, one of the shops I worked at used it, and one of my friends was an expert at generating test scripts, but we were a software shop. Our product *was* the software, making it much more critical that there be as close to zero defects as possible. That’s not the case if your product is not software. In those cases, software is a supporting tool to help sell product, and as such the occasional bug, while regrettable and to be avoided, doesn’t reflect upon your entire product line.

So there are some tools out there to do testing. They’ll help determine things like buffer overflows and such, and Microsoft is selling Microsoft Test Center, which looks to be a more comprehensive solution, but no program knows your business – and thus the testing context – in the manner that your employees do, and most companies are not willing to shell out the money required to start a dedicated testing group. If you are one of the lucky ones, well you can replicate your entire production environment with VMs and an ADC… But you can’t get the volumes you would see in a live public-facing Internet application with actual personal interaction and all the dumb mistakes we users make. So you can test, after a fashion, and with real people involved even set up intelligent testing, but it will take some effort. The best bet for most shops is to make unit testing part of the developers’ jobs, and system testing part of the project managers’ job, with help from both dev and ops. See, Devops!

It is a testament to the quality of tools and developers out there that we don’t see a ton of issues all the time. Think if the Web had the percentage of problems on a daily basis that Windows Apps did early on… We’d never get anything done. So don’t take DevOps so seriously from a technological standpoint, instead, let’s talk about culture.

Developers need to feel like part of the larger team if you want them to worry about DevOps. Here’s the catch, most of them don’t want to feel like part of the larger team. Network issues and storage shortages annoy them as much as your business users. They want to write cool apps and shove them out the door (or the Internet connection, more precisely) without worrying about deployment issues too much.

It is up to management to take concrete steps to move dev closer to ops. To do this, first mandate that all testing occur on the OS that deployment will occur on. This shouldn’t be a big deal for most shops, but in a few it will be. Second, mandate that the dev team put resources on the deployment, and utilize the Virtualization/ADC scenario discussed above. Third, remind developers that their app isn’t great unless users think it is, and it is deployable. They’ll come around, but it will take some work on your part.

The days of developing in a vacuum are long gone (at least in the enterprise), and the disconnect between dev and ops is one of the few remaining artifacts of that time. You just have to remove the artifacts and hook up the strengths of your ops team with the leet app skills on your dev team, and the whole concept of DevOps is significantly reduced.

One last bit, troubleshooting when things do go wrong. Developers can build a lot into their apps to give them hints about status, but ops can use tools like iApps (built into TMOS v 11 on F5 BIG-IP LTM) to show that it is indeed the application not responding in a timely manner – or indeed to show exactly what IS the bottleneck in a deployment. The reporting functionality on iApps makes them a worthwhile endeavor without the “ease of infrastructure management” they offer. iApp reporting can tell you exactly which piece of the application environment is slow, dragging the entire system response time down. I think that’s huge. If you have a BIG-IP, check it out, I’m pretty certain you will too.

So call a meeting. Make it around lunch time and announce that pizza will be provided. Both teams will show up, and you can start changing culture. The benefits will be long-term, with applications better suiting users needs and requiring less operations man-hours. And devs will get a better feel for what works and doesn’t in your environment.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com