Decoding DevOps

It was in the 60s that the operations concept actually started, when software engineers wearing a geeky lab coat worked on vacuum tubes. From there, it was a shift to mounting tapes to IBM JCL's where operations teams started identifying themselves as a non-code writing team. But, it was in the 1980's and 90's that modern "IT Operations" culture was born. This is the era when PCP users needed support, resource sharing, networks, etc. And then a wall was built between developers and ops team for numerous reasons. In the era of Cloud computing, the ops team seems to have gone obsolete but actually the operations don't go away. Responsibilities can, and do, shift over time, and as they shift, so do job descriptions. But, no matter how you slice it, the same jobs need to be done, and one of those jobs is operations.

(Source: http://blog.zenoss.com/)

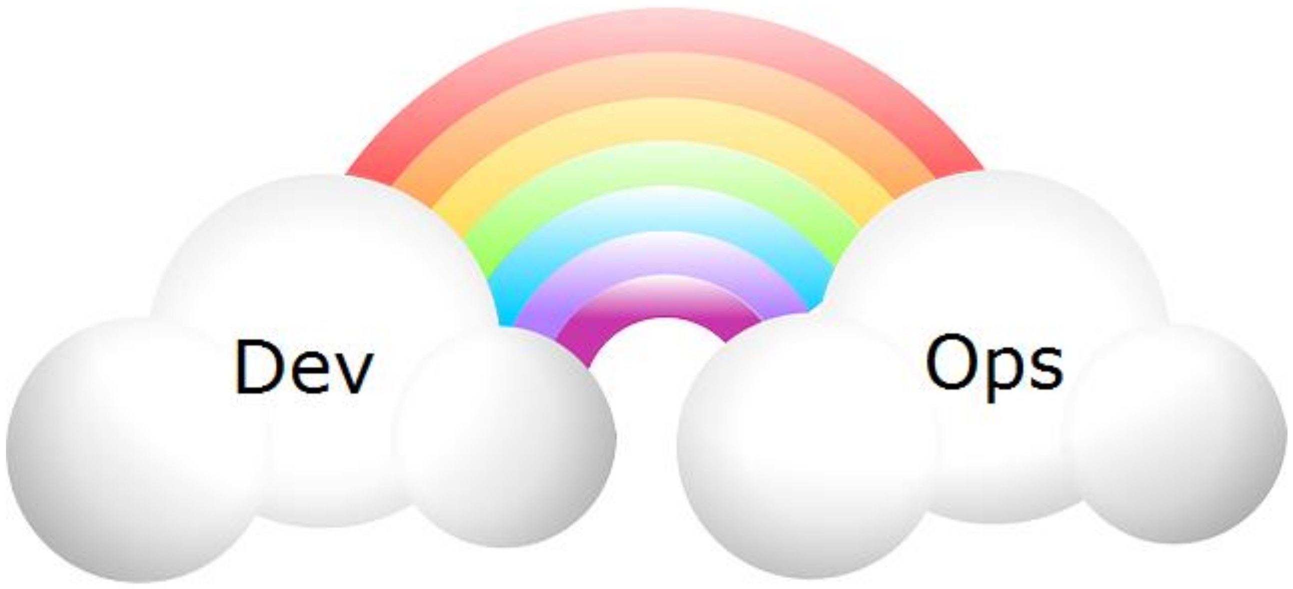
The DevOps culture is trying to find the answer to the same questions that are making ops obsolete. DevOps represents a shift in thinking and culture. Instead of regarding software development and IT operations as separate entities, the DevOps perspective considers them as areas of technical responsibility, taken on by people who work closely and collaboratively towards the same purpose: faster release of software that fulfills business requirements and goals.

Infrastructure as a code

In the current heavily virtualized world of infrastructure, it's impossible to maintain it without using a proper tool. So, the job profile of the Operations team has changed – people who never used to code, now have perl scripting and ruby scripting as part of their resumes. It has given system admins a new way of maintaining the infrastructure-as-a-code, check it into a version control system and keeps it updated with configuration changes. This has made processes like continuous integration very easy but has also brought wrong expectations that we can make 10 releases a day.

Use of Cloud in DevOps

One important task of operations is understanding the cost trade-offs between public clouds like Amazon's AWS, private clouds, traditional co-location, and building their own infrastructure. It's hard to beat Amazon if you're a startup trying to conserve cash and need to allocate or de-allocate hardware to respond to fluctuations in load. You don't want to own a huge cluster to handle your peak capacity and leave it idle most of the time. But Amazon isn't inexpensive, and a large company can probably get a good deal taking its infrastructure to a co-location facility. A few of the largest companies will build their own data centers. Cost versus flexibility is an important trade-off; scaling is inherently slow when you own physical hardware, and when you build your data centers to handle peak loads, your facility is underutilized most of the time. Smaller companies will develop hybrid strategies, with parts of the infrastructure hosted on public clouds like AWS or Rackspace, another part running on private hosting services, and yet another part running in-house. Optimizing how tasks are distributed between these facilities isn't simple; that is the province of operations groups. Developing applications that run effectively in a hybrid environment: that's the responsibility of developers, with healthy cooperation from operations team.



(Source: www.dbmaestro.com)

Monitoring in DevOps

Early system monitoring tools like HP's OpenView provided limited visibility into the system and network behavior but didn't give much information than simple heartbeats or reach ability tests. Modern tools like AppDynamics provide insight into almost every aspect of system behavior; one of the biggest challenges facing modern operations groups is developing analytic tools and metrics that can take advantage of the data that's available to predict problems before they become outages. We now have access to the data we need; we just don't know how to use it. And the more we rely on distributed systems, the more important monitoring becomes. As with so many other things, monitoring needs to become part of the application itself. Operations are crucial to success, but operations can only succeed to the extent that they collaborate with developers and participate in the development of applications that can monitor and heal themselves.

Contributed by Ravi Petlur.

Visit us at Neevtech.com to know more about our offerings.

Please visit our blog for more.