APM Has to Change to Support DevOps

Published: 23 March 2016 **ID:** G00281276

Analyst(s): Cameron Haight, David Paul Williams

Summary

The APM status quo is not sufficient to support an effective DevOps initiative. I&O leaders implementing DevOps will need to not only rethink the tooling, but also address people and process (including information flow) concerns in order to better enable the business.

Overview

Key Challenges

Close coordination among IT stakeholders is lacking today, inhibiting the ability to deliver more effective application performance management (APM) services.

A traditional IT operations/APM organizational structure is not optimized to support DevOps-driven digital business initiatives.

A key DevOps requirement that technologies in support of processes such as continuous delivery including monitoring (APM) be part of an integrated and automated toolchain is often still a work in progress.

Few organizations with APM tools take a holistic approach to measurement of key performance indicators (KPIs) in their DevOps programs.

Recommendations

Implement a behavior-driven development (BDD) approach to ensure that your APM tooling is able to provide the right information to the right IT audience.

Develop a site reliability engineering (SRE) team to enable more rapid and customized delivery of APM-related and other monitoring information.

Require that your APM technology support open APIs that enable the receipt and transmission of events from and to other components within a DevOps environment.

Use APM technology to measure performance and to support KPIs across the DevOps toolchain life cycle.

Introduction

DevOps is forcing tremendous change on IT organizations that are implementing it to improve support of the business. Most of the attention, with respect to DevOps, has been on improving product time to market (or deployment), which has caused significant change in the processes and tooling in support of this need. IT organizations now need to focus on amplifying part of the feedback loop (what is often referred to as the "second way" within the DevOps community) by rethinking not only their APM tools, but also the related processes, information flows, and organizational structure, thereby implementing and supporting a DevOps initiative to rapidly identify problems and implement their remediation.

Analysis

Develop an SRE Team

There are several challenges with the existing organizational approach to APM that must be addressed:

- The DevOps philosophy emphasizes the breaking down of silos; yet in traditional enterprises, IT operations organizations still often have dedicated application (not to mention server, network and storage) administrative teams focusing on their own needs and tooling.
- The increasing breadth of the information needs of the consumers of APM and other monitoring data will likely require some degree of customization of deployed commercial software offerings. This may be as simple as setting up additional dashboards and monitors to instrumenting the application code itself.
- There still remain motivational differences between development and operations teams, which are somewhat reinforced by APM data knowledge disparity, causing conflicts with respect to whether or not to ship new application updates.

Recommendations

- Develop a team that has responsibility for life-cycle-oriented and cross-cutting application (and infrastructure) data collection concerns, and that will work closely with software engineers and other key IT personnel (see "Principles and Practices of DevOps"). In large cloud organizations, this team is often called the SRE team, and has responsibility for implementing platforms for monitoring and automation. Note that this approach addresses the problem of a lack of context when, for example, looking at data from only one technology domain. In addition, this reduces the dependency problems with key personnel that only know their view of the "system," which can create challenges with organizational scalability.

Ensure that your SREs are programmer-level proficient (they may, in fact, also be candidates for regular software engineering [development] positions if they so desire). Hence, there should be parity in terms of capabilities to ensure mutual respect between development and operations. In addition, they should spend, on average, 50% of their time providing extensions to existing tooling (if not actually building their own monitoring technology), as well as enabling automation (to reduce manual interventions and improve mean time to failure [MTTF]/mean time to repair [MTTR]). Self-service capabilities for the increasing number of monitoring consumers outside of IT operations should also be a priority. While enterprises may struggle to find similar skills, they should aim, where possible, for the same capabilities.

Provide a neutral enforcement mechanism to the delivery of new application functionality. The concept of error budgets that are defined by development-driven service-level objectives (SLOs), provides an unbiased means of assessing whether or not the delivery of new application functionality is permitted. Once APM-measured availability (or performance), for example, falls below the acceptable budget threshold, the SRE team is empowered to defer all updates until stability returns to the system.

Implement a Behavior-Driven Development Approach

The lack of formal structures for the exchange of information between consumers of APM and monitoring information and the providers of that data reduces IT operations effectiveness and reputation. Behavior-driven development, an approach used in software development (see "Increase Collaboration and Drive Agility With Behavior-Driven Development"), should be considered for use in an operations context to address the key challenges below:

A lack of communication between software engineering and line-of-business organizations and the IT operations teams tasked with meeting their monitoring and informational needs. This can impact time to market, as well as knowing when a requirement is "done."

Application monitoring specifications and requirements are usually not treated as first-class artifacts (or key documentation), because there is no direct accountability and, thus, there is often little ability to build upon past development efforts by the SRE team and other IT operations teams.

Recommendations

SREs and other individuals responsible for APM and monitoring within IT, if not embedded in Scrum or other agile development teams, should hold regularly scheduled discovery workshops with these teams (for more information on Scrum see "Making

Sense of the Agile Methodology Wars"). The goal of these workshops would be to develop a gap analysis between desired versus delivered APM data. Ideally, this should occur before the start of a new Scrum sprint or development effort.

Use a tool (like Cucumber or RSpec) to provide a consistent syntax in a human-readable format for software engineering — and even nontechnical — teams to express desired APM and other monitoring features and, thus, reduce ambiguity. SRE teams would then create step definitions (or the code or configurations necessary to execute the scenario as defined in the feature file) from this input to test for the desired APM system functionality. Note that SRE and other monitoring teams should also consider using the same BDD/test-driven development (TDD) techniques to test the correctness of deployed applications and infrastructure. (For more information on behavior-driven development, see "Increase Collaboration and Drive Agility With Behavior-Driven Development.")

Require That Your APM Technology Supports Open APIs

Your APM technology should support open APIs that enable the receipt and transmission of events to and from other components within a DevOps environment. Context is king in DevOps; yet there are often impediments that prevent the ability to collect and share information among differing toolsets, for example:

APM products may lack consistent, RESTful APIs to enable the efficient collection of information from a broad portfolio of other APM, log management and configuration management tools.

Without a well-defined and integrated toolchain, collaboration among your SRE team and the team's development workflow will likely suffer due to the inability to fully automate the APM feedback loop.

Recommendations

Use an API definition to drive contract-first (or implementation-independent) monitoring API design that helps to reinforce the separation of the APM product interface from the resources it exposes, reducing monitoring lock-in and maximizing flexibility.

Leverage newer capabilities in technologies (such as Swagger or RESTful API modeling language [RAML]) to define the functionality of a RESTful API, and provide automatic generation of APM APIs as well as API documentation (see "Guidance Framework for Creating Your API Developer Toolbox").

Use APM Technology to Measure Performance and to Support KPIs

Performance should not solely be the domain of the IT personnel monitoring the live environment using APM tools to monitor for application performance issues. An effective DevOps practice will understand and capture business performance expectations, and track these expectations through development and into production. This is accomplished by using APM technology throughout the DevOps process in a consistent manner (to ensure performance is measured using the same algorithms, methods and metrics). The following example explains how APM contributes to establishing a performance practice through the ongoing DevOps cycle:

1. Establish the business performance objective
2. Code against objective
3. Test against objective
4. Establish performance test metric
5. Perform canary or incremental releases into live environment
6. Measure performance deltas (drift) in live environment against test metric
7. Factor drift from business and test objectives
8. Identify source of the performance drift
9. Go to step 2

Recommendations

Choose APM technology that provides the necessary (and consistent) level of data visibility that is of value throughout the DevOps cycle. Examples of the types of data that should be considered include:

Component diagnostics provides DevOps application development teams with a deep diagnostic technology for understanding infrastructure component performance. This includes the ability to trace function calls and application components, and to find bottlenecks and potential root causes in the code. For DevOps, a key value is when this technology is used by developers to build the monitoring and measurement capabilities into the application — prior to deployment.

Transaction tracing provides a view into application transaction routes, which can include a topological dependency mapping of the application (infrastructure, middleware and application layers), and tracing of end-user as well as back-end transactions. This is critical for isolating issues, such as latency to a specific area or domain.

End-user experience managers provide a view into end-user activities from the IT Web tier. Requests are intercepted, analyzed and reconstructed back into full user sessions and transactions. Dynamic round-trip baselines allow performance degradation to be identified, capturing the data needed to diagnose the cause and remediate the issue with priorities established based on who is affected, where they are and what the impact is.
