WHITE PAPER | NOVEMBER 2016
DEVOPS PRACTITIONER SERIES

Shift-Left Monitoring Strategies for Agile Operations

Integrating Application Performance Management with Jenkins Continuous Integration

Srikant Noorani
Senior Engineering Architect

ca® technologies

Table of Contents

Executive Summary

Challenge

As organizations accelerate digital transformation strategies, the pressure mounts on IT to support an increased variety of customer-centric applications. And with agile practices and continuous delivery fueling constant change, ensuring a high-quality customer experience becomes increasingly complex. The traditional model of engaging IT operations to take ownership and responsibility for quality when applications are deployed to production is no longer viable or sustainable. Not only does it increase conflict and slow delivery, it severely damages the business when undetected software defects lead to a suboptimal customer experience.

Opportunity

Progressive organizations recognize that establishing high-quality customer experiences can't wait until production. By employing DevOps practices and automation, cross-functional teams collaborate and share performance information across the software lifecycle to build quality into applications as they're rapidly designed, developed, tested and deployed. A perfect opportunity to support this goal presents itself during continuous integration (CI). By shifting left application performance management (APM) from production into the CI process, cross-functional teams can catch problems early, where they are less costly to fix.

Benefits

Integrating APM with CI tools such as Jenkins helps teams establish quality without compromising velocity. APM exposes rich information in context of build processes, meaning developers gain clear visibility into the impact of their code on performance and can immediately pinpoint problem root cause. With a single source of the truth, cross-functional teams collaborate toward meeting the shared goal of delivering and enhancing a high-quality customer experience at a faster pace. This key integration helps organizations realize numerous DevOps benefits, including:

- Fast feedback between development and operations
- Increased application resilience from early, proactive monitoring
- Improved application supportability and lower operations costs
- Increased productivity—developers spending less time firefighting production problems

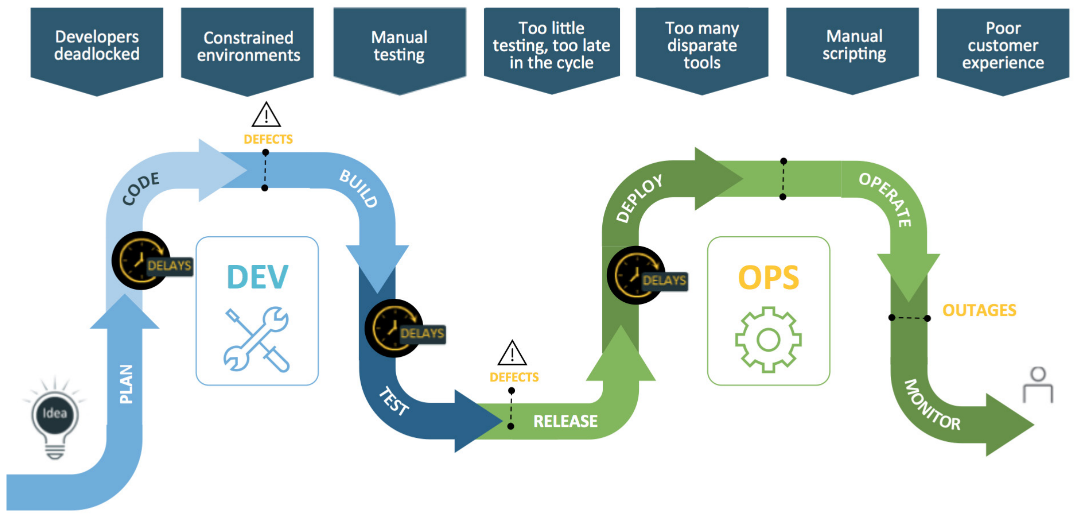
Section 1

Imperative: Faster Speed and Increased Quality

Historically, application software has been delivered in long cycles using waterfall-style development methods. In such situations, strict organizational demarcation has left IT operations being employed late in the development cycle to ensure that the software delivered performs in accordance with service level agreements. Because application performance quality is retrofitted when software is passed "over the wall," delivery can stall as operations conducts important checks, or worse still, defects go undetected and cause outages or a suboptimal customer experience (see Figure 1).

Figure 1.

Separation of development and operations (people, process and technology) slows application delivery and impacts quality.



Significantly, this model may adversely impact organizational culture. By working in discrete silos where developers are focused on speed and operations on performance quality, low-trust working practices and conflict may persist. In severe cases, IT operations enforces inflexible change management and standardization demands, while development looks to bypass existing controls to support faster delivery.

With agile methods now supporting rapid development, establishing quality at the end of cycles is no longer sustainable. Iterative-based software development and automated continuous integration and delivery processes mean software can potentially be deployed much faster. The software delivered is changing radically too. In addition to delivering systems to support internal business processes, development is now tasked with delivering customer-centric applications that support multichannel engagement. With these styles of applications, success depends on being able to support rapid functional change, while delivering a high-quality customer experience—the two are inseparable.

Studies have shown that high-performance IT organizations are consistently outperforming their lower-performing peers in terms of velocity. According to the 2016 State of DevOps report presented by Puppet Labs and DevOps Research and Assessment (DORA), high performers deploy software 200 times more frequently than low performers, with 2,555 faster lead times.[1]

But the study also illustrates that velocity doesn't need to compromise quality. Even at increased speed, high-performing IT organizations recover 24 times faster and have 3 times lower change rate failures.[2]

DevOps as a transformational change agent

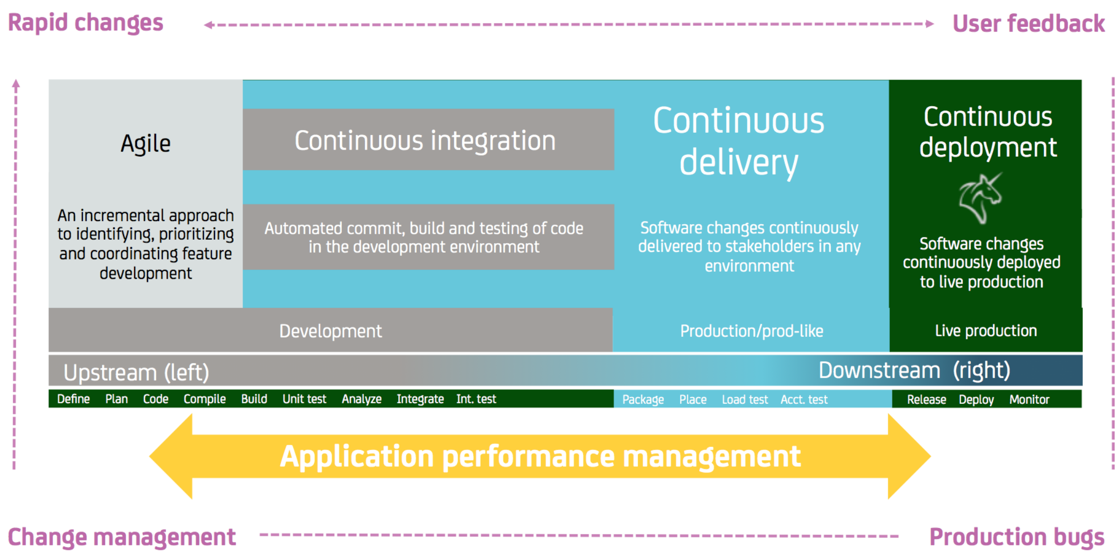
DevOps is consistently helping organizations realize speed and quality goals, but success requires a high-trust culture fueled by cross-functional team collaboration. This was well illustrated in the "Assembling the DevOps Jigsaw" survey, which recognized the importance of collaboration, but showed that of the organizations conducting DevOps initiatives, only 35 percent of have established cross-functional IT processes, with even fewer (29 percent) gaining complete cultural harmony within IT.[3]

Clearly, and as the report suggests, establishing cross-functional processes has been problematic when teams work in silos using their own tools. But with speed and quality inseparable, methods must be pursued whereby integrated tools allows teams to coordinate efforts around common business goals. This should occur early within the development lifecycle, where, through process integration, cross-functional teams share and benefit from each other's information to ensure higher levels of quality are systematically "baked" into applications—as they're coded, built, tested and deployed.

As illustrated in Figure 2, there are many areas where quality can be established by using a shift-left monitoring approach. One example is with the CI process. Here, developers leverage open source tools like Jenkins to automate source code commits, builds and testing in development environments. This provides an excellent point for APM (a practice normally reserved for production environments) to be leveraged by developers to ensure that what they're building meets production performance needs—well before

Figure 2.

Shift-Left APM


Rapid changes
User feedback
Agile
An incremental approach to identifying, prioritizing and coordinating feature development
Continuous integration
Automated commit, build and testing of code in the development environment
Continuous delivery
Software changes continuously delivered to stakeholders in any environment
Continuous deployment
Software changes continuously deployed to live production
Development
Production/prod-like
Live production
Upstream (left)
Downstream (right)
Define Plan Code Compile Build Unit test Analyze Integrate Int. test
Package Place Load test Acct. test
Release Deploy Monitor
Application performance management
Change management
Production bugs

deployment.

Essentially, the notion of utilizing APM in preproduction with CI should be considered a DevOps best practice; however, this must support a number of key requirements, including:

- Serving developers performance information in context of the immediate build process
- Providing automated mechanisms to fail software builds if performance KPIs are not met
- Delivering easy-to-use methods to pinpoint problem root cause

- Ensuring bidirectional information flow—for example, publishing and visualizing CI information in APM

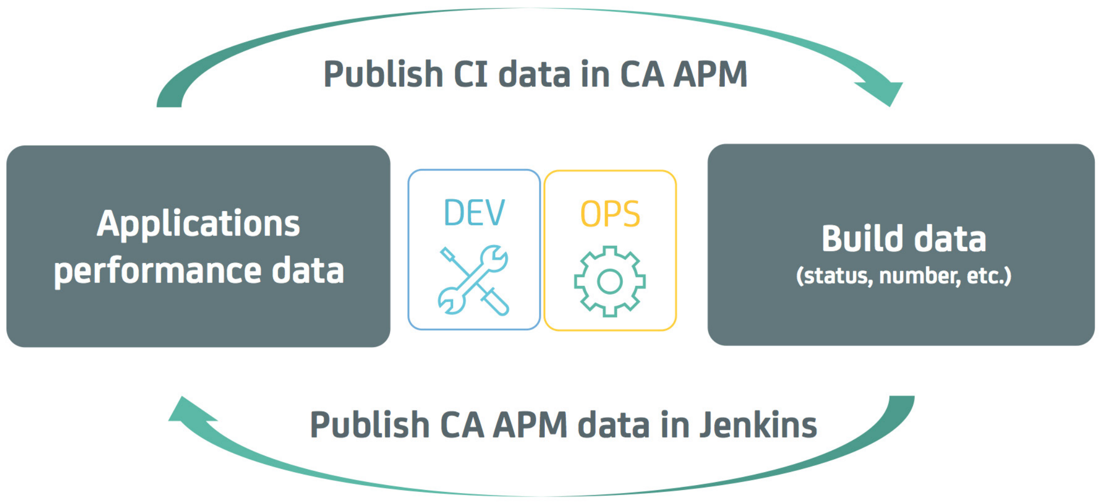
Section 2

Shift Quality Left—APM and Jenkins Integration

By integrating APM with CI, cross-functional teams can purposefully collaborate toward exceeding speed and quality goals. Using the integration between CA Application Performance Management (CA APM) and Jenkins as an illustrative example (see Figure 3), this capability ensures that developers have immediate access to application performance data (published to Jenkins) during the build process, while Jenkins attributes, such as build number and status, are surfaced to APM.[4]

Figure 3.

Bringing APM and CI together


Publish CI data in CA APM
Applications performance data
DEV
OPS
Build data
(status, number, etc.)
Publish CA APM data in Jenkins

This bidirectional flow of information is important because it gives both sets of practitioners access to critical metrics in the context of their work and the tools they normally use. For example, developers gain immediate access from the Jenkins screens to application performance information, without having to switch to another tool, while operations and application support teams don't need to familiarize themselves with the all intricacies of Jenkins.

The nature of the CA APM-Jenkins integration also helps support another key goal of DevOps—information sharing. By automatically exposing application performance metrics normally reserved for production, developers gain immediate visibility into the effects of their software builds from a quality perspective. Again, sharing is bidirectional, with APM administrators instantly knowing how the build performed without the need to access other tools or reports.

Functional overview, capabilities and benefits

The CA APM-Jenkins integration delivers the following core functions and capabilities:

- **In-context visualization.** During the build process, developers have immediate access to relevant APM

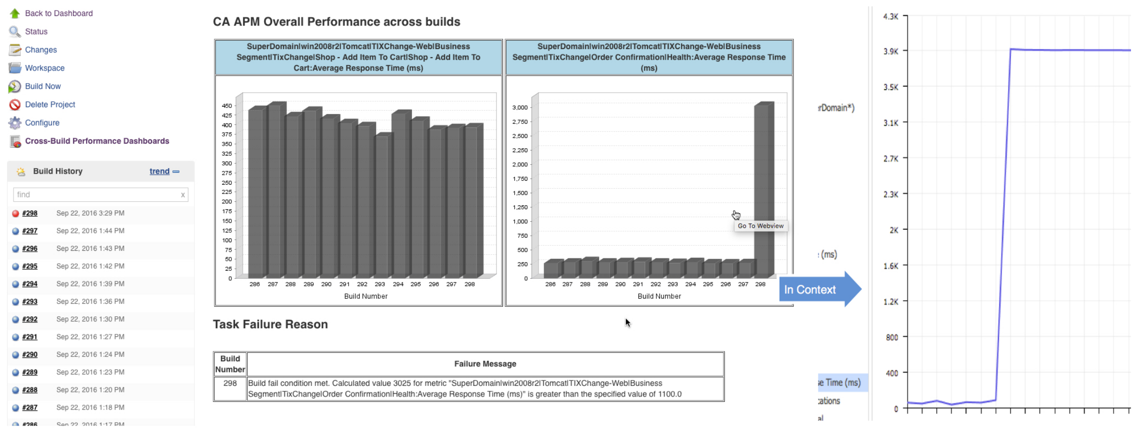
performance information directly from Jenkins.
Benefit. No lengthy task-switching to unfamiliar tools: By presenting information in context, developers gain visibility into the performance impact of their work, together with the identification of problem root cause.

- **Detailed application performance information.** Multiple preproduction APM metrics can be collected and delivered to Jenkins as KPIs during the build processes.
  Benefit. Allows developers to immediately assess the efficacy of their code against metrics they don't normally have access too; this saves time and increases both quality and productivity.

- **Multiple build comparisons.** Developers can instantly see quality impacts between their builds (see Figure 4).

Figure 4.

Multiple build comparisons with APM information presented in the context of work conducted in Jenkins.


Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Cross-Build Performance Dashboards
Build History
trend
find
CA APM Overall Performance across builds
SuperDomain|win2008r2|Tomcat|TIXChange-Web|Business Segment|TixChange|Shop - Add Item To Cart|Shop - Add Item To Cart:Average Response Time (ms)
SuperDomain|win2008r2|Tomcat|TIXChange-Web|Business Segment|TixChange|Order Confirmation|Health:Average Response Time (ms)
Build Number
Go To Webview
In Context

#298	Sep 22, 2016 3:29 PM
#297	Sep 22, 2016 1:44 PM
#296	Sep 22, 2016 1:43 PM
#295	Sep 22, 2016 1:42 PM
#294	Sep 22, 2016 1:39 PM
#293	Sep 22, 2016 1:36 PM
#292	Sep 22, 2016 1:30 PM
#291	Sep 22, 2016 1:27 PM
#290	Sep 22, 2016 1:24 PM
#289	Sep 22, 2016 1:23 PM
#288	Sep 22, 2016 1:20 PM
#287	Sep 22, 2016 1:18 PM
#286	Sep 22, 2016 1:17 PM

Task Failure Reason

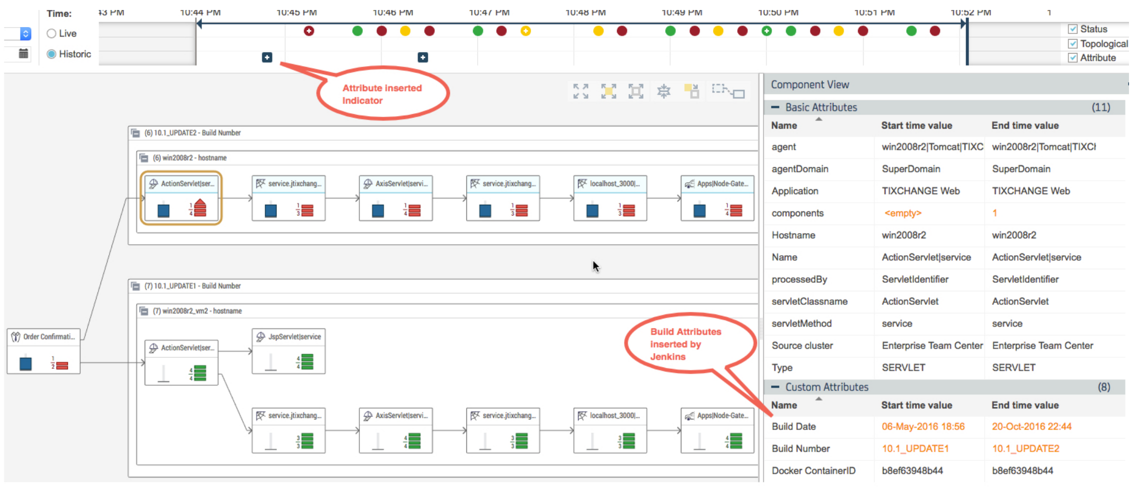
Build Number	Failure Message
298	Build fail condition met. Calculated value 3025 for metric "SuperDomain|win2008r2|Tomcat|TIXChange-Web|Business Segment|TixChange|Order Confirmation|Health:Average Response Time (ms)" is greater than the specified value of 1100.0

Benefit. Any code-related issues being introduced that could have a negative impact on quality are identified early in the development process; this reduces technical debt and helps avoid problems leaking into production where they are more costly to address.

- **Automate build/fail conditions.** With the ability to automatically mark any build as pass or fail based on configured conditions, developers can troubleshoot performance issues by clicking on the cross-build performance dashboard in Jenkins and jumping to APM (again, in context) to conduct detailed root-cause analysis.
  Benefit. Faster triage of problems before production and increased levels of code base quality. When a build fails, the developer responsible for that build is the one who addresses the problem.

- **Streamlined build process.** Key applications performance checks are established and automated within CI. Rather than rely on manual processes, increasing quality becomes a key component of the build process and associated workflows.
  Benefit. Developer productivity increases because they can avoid lengthy delays in conducting manual application performance checks against each software build.

- **Establishes strong feedback loops.** Key build-related information published to APM enables application support staff to instantly know how the build performed, without referring to Jenkins.
  Benefit. Communication strengthens across teams and reduces training costs. Once again, information is delivered in context of the practitioner's role (in this case APM support administrators—see Figure 5).



Component View

Basic Attributes (11)

Name	Start time value	End time value
agent	win2008r2|Tomcat|TIXC	win2008r2|Tomcat|TIXCI
agentDomain	SuperDomain	SuperDomain
Application	TIXCHANGE Web	TIXCHANGE Web
components	<empty>	1
Hostname	win2008r2	win2008r2
Name	ActionServlet|service	ActionServlet|service
processedBy	ServletIdentifier	ServletIdentifier
servletClassname	ActionServlet	ActionServlet
servletMethod	service	service
Source cluster	Enterprise Team Center	Enterprise Team Center
Type	SERVLET	SERVLET

Custom Attributes (8)

Name	Start time value	End time value
Build Date	06-May-2016 18:56	20-Oct-2016 22:44
Build Number	10.1_UPDATE1	10.1_UPDATE2
Docker ContainerID	b8ef63948b44	b8ef63948b44

Figure 5.

Jenkins build attributes are automatically inserted by Jenkins in CA APM and visible from the Team Center dashboard.

Shift-left use case: integrated workflow

The power of the CA APM-Jenkins integration is well-illustrated in Figure 6 below. As Jenkins automates the build process, the integration with CA APM helps DevOps practitioners establish quality (without impacting delivery speed) within the release before production. The integration helps foster better communication within and across teams so that speed and application quality become shared goals.

As Figure 6 shows, the workflow starts with a developer checking in code with new functionality. Jenkins then conducts the build process and automates tests. This automation includes accessing performance data via the CA APM-Jenkins integration, which is made available to Jenkins.

At this stage, Jenkins will mark the build as pass or fail based on preconfigured conditions. The integration automatically generates a cross-build dashboard and presents it in context to the developer, who doesn't have to access an unfamiliar tool to see the information.

If the build has failed a condition, the developer can immediately observe quality impacts between builds. Using the dashboard, the developer can then pinpoint the exact cause of the problem from the code constituting the latest build. This can include an actual method or line of code.

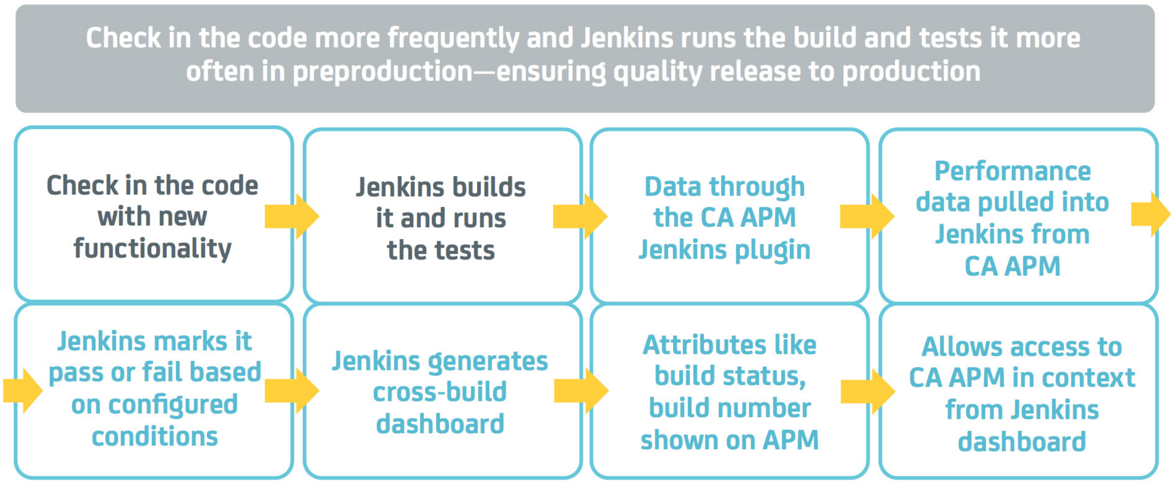
Since the integration includes email notification, a message outlining a build fail condition together with a reason (for example, number of method calls greater than specified threshold) is emailed to the developer. This helps teams further identify the exact reason for the problem and immediately conduct more targeted triage.

The integration ensures that information also flows to CA APM so that application administrators can review attributes such as build status and number from the CA APM dashboards which they're accustomed to working with.

An additional benefit from this integration is how it helps both developers and IT operations increase knowledge about an application as it progresses toward production. Armed with CA APM data, developers gain more insights into the effect of their code on application quality. As their knowledge increases, developers can avoid detrimental coding practices, while repeatingtechniques that have had a positive impact on application quality. For IT operations, this knowledge capture helps build an early picture of operational support and service-level requirements before an application reaches production status.

Figure 6.

CA APM-Jenkins: integrated workflow


Check in the code more frequently and Jenkins runs the build and tests it more often in preproduction—ensuring quality release to production
Check in the code with new functionality
Jenkins builds it and runs the tests
Data through the CA APM Jenkins plugin
Performance data pulled into Jenkins from CA APM
Jenkins marks it pass or fail based on configured conditions
Jenkins generates cross-build dashboard
Attributes like build status, build number shown on APM
Allows access to CA APM in context from Jenkins dashboard

Section 3

Summary: Proven Value of Moving APM to Preproduction

Progressive organizations recognize that application quality doesn't start and end in production. For them, quality is built into applications early in the development cycle so that problems can be identified and fixed before they adversely impact the business.

With practices such as agile, continuous integration and delivery supporting faster deployments and shorter lead times, failing to establish quality early can and will incur significant business costs.

By moving APM into preproduction, organizations can realize increased value from their DevOps initiatives, including:

- **Teamwork, collaboration and trust.** Because quality is established and demonstrated during the development cycles, a high-trust culture develops. IT operations and application support staff trusts what development is delivering is of high quality, while development trusts operational monitoring because it helps them produce better code.
- **Increased productivity across teams.** By identifying and fixing problems during software build processes, developers spend less time working on serious production problems and more time writing great code. This productivity benefit was illustrated in a Forrester Research report, which indicated a 15 percent improvement in application delivery cycles when APM was established in pre-production[5] . This increased productivity can have a positive impact on the business, as indicated by one customer, "In essence, we're issuing less code or less fixes into production, which allows [us] to focus more on maintaining a competitive advantage in the marketplace of what we deliver in our online solution."[6]
- **Faster problem resolution.** APM and the methods described in this paper can also accelerate problem resolution times (as much as 90 percent according to the Forrester report[7]). By presenting APM data to developers in the context of their Jenkins work and automatically passing/failing builds, staff can quickly and efficiently identify problem root cause.

Section 4:

About the Author

Srikant Noorani is a senior technical architect with CA Technologies based out of Toronto, Canada. With more than 18 years of experience in the IT sector, Srikant has worked for both startups and large technology providers, including Sun Microsystems and Fujitsu. Before joining CA, Srikant helped with one of the largest implementations of application performance management at Blackberry.

1 Puppet + DORA, "2016 State of DevOps Report," March 2016

2 Ibid

3 Freeform Dynamics, "Assembling the DevOps Jigsaw," 2015

4 Jenkins Wiki, "CA APM Plugin," Mar 15, 2016

5 Forrester Consulting, "The Total Economic Impact™ of Application Performance Management from CA Technologies," April 2015

6 Ibid

7 Ibid