DevOps Best Practices: Combine Coding with Collaboration
(Part Two of a Two-Part Series)

Effectively merging application development and operations requires organizations to assess what they do best and seek partners who can deliver greater operational agility.

Executive Summary
In part one of this series, we described how DevOps requires not only new ways of working but also new ways of thinking. Rather than focus only on the new features or interface elements they designed, developers must work with the operations teams to package, test, deploy and manage those features. Rather than roll their eyes at the threat to uptime and add a new code release to next week’s testing schedule, the operations team must trust automated scripts to test and roll out new code in hours or even minutes.

To drive the maximum business benefit, the “Dev” and “Ops” teams must work together to continually monitor machine and user data, fix any problems and test and deploy new code. In many organizations, “Dev” and “Ops” roles are already merging, and DevOps is helping bridge these two worlds. DevOps provides the ability to write scripts that automate deployment, monitoring and test processes that used to require lengthy manual effort. It also provides the communication and change management skills to help historically separate development and operations teams understand each other’s needs and work together to meet, in real time, the ever-changing needs of users, customers and business partners.

Implementing DevOps most quickly and cost-effectively also requires understanding what the organization itself can do best, and what parts of the DevOps process to leave to partners with more experience.

DevOps Requirements
DevOps collapses previously separate development and operations into combined functions, with a “fail fast” mentality of rapid deployment, rapid learning and rapid course corrections.

Change of this magnitude is always a challenge. When Puppet Labs and IT Revolution Press surveyed more than 4,000 developers and IT operations professionals, they found the most common difficulties in implementing DevOps were:

- The value of DevOps was not understood outside their group (48%).
- The lack of a common management structure for development and operations (43%).
- A lack of DevOps tools (33%).
• Lack of time to implement DevOps (31%).
• Lack of support (19%).

While those who had no plans to implement DevOps cited:
• Lack of manager buy-in (49%).
• Lack of team buy-in (38%).
• Budget constraints (19%).
• More hype than substance (14%).

Given the need to change workflows and share responsibilities in the DevOps model, “soft” skills such as communication and collaboration are required as well as more technical capabilities such as coding. This is borne out by the Puppet Labs/IT Revolution Press survey, which found the skills respondents needed most to implement DevOps to be:

• Coding/scripting (84%).
• People skills (60%).
• Process reengineering skills (56%).
• Experience with specific DevOps tools (19%).

Coding and scripting skills are required to automate manual processes with code that can be reused across multiple cycles of automated code testing and deployment. This representation of the IT infrastructure as code increases the consistency of the IT environment, while the automation it provides reduces deployment delays and increases business agility.

Once new code has been checked into a version control repository, these scripts submit the code for testing through a set of automated frameworks. If the code passes the required tests, another script converts it into a build for deployment. If it fails to pass, the script can send the code to an additional "sandbox" for testing before deployment.

Interestingly, the survey respondents gave relatively low priority to their staff having experience with specific DevOps tools. This mirrors our experience. We have found that skills in specific tools are more easily taught than, for example, “soft” skills such as the ability to resolve disputes between the development and operation staffs.

The tools that survey respondents found most important to their DevOps efforts were:

• Version control (84%).
• Configuration management (78%).
• Ticketing (68%).
• Resource monitoring (60%).
• Provisioning (56%).

A centralized version control platform provides a single, trusted source of both new code and the scripts that configure the infrastructure to run it. Configuration management tools implement the scripts in an automated, consistent way to speed time-to-market and assure consistency for service quality. Ticketing systems help the combined development and operations staffs track the status of pending enhancements in the infrastructure, or in digital products, applications or services.

Automated and intuitive resource monitoring provides real-time visibility into service quality issues and helps DevOps quickly or even proactively resolve them. Provisioning platforms make the real-time changes in the server, storage or network infrastructure required to implement new features or assure compliance with service level agreements.

**Processes and Workflows**

A great DevOps platform boils down to four key enablers:

• Intensive tooling.
• Tool chain integration.
• Workflow automation.
• Self-service enablement via real-time and on-demand reports and/or insights.

Along with tools and technology, our work with clients has identified critical processes and workflows required to get the most rapid and significant benefits from DevOps.

Some of the most important are the identification, deployment and integration of a common DevOps toolkit. This should span the entire cycle from code development through testing to packaging, deployment and monitoring. This comprehensiveness provides a complete view of the entire environment, the changes required in it, the status of those changes and any bottlenecks to making those changes.
This toolkit must be common across all the staff involved in DevOps to minimize training and licensing costs. Even more importantly, a common toolkit assures all the involved staff can share best practices, understand each other’s challenges and together create and comply with best practices.

Automation is another factor essential to DevOps success. Automation obviously significantly reduces costs by reducing manual effort. But the real benefits come through greater agility and the ability to more quickly fix problems or enhance applications or services in response to business needs. Automated, policy-based processes also help ensure consistency, which in turn prevents misconfigurations that can endanger uptime, performance and compliance.

Organizations also need to identify, collect and analyze both machine and end-user data. This may involve new workflows and tools to collect and integrate information from what historically were data silos. Some organizations, however, may be able to draw from toolsets and best practices already in place for their data warehousing and business intelligence initiatives. These may include tools to extract, transform and load data and to provide appropriate security for the merged data.

Finally, organizations must implement end-to-end management and oversight to ensure that their slowest processes don’t delay deployment. A seamless process of release management, unit builds, unit test, promotion and deployment helps prevent last-minute surprises during integration. This may require new workflows, assignment of responsibilities and oversight and audit processes.

Sourcing Choices

As we described earlier, the value of DevOps comes through greater agility and service quality, not from developing your DevOps infrastructure. For that reason, if your organization is convinced of the benefits of DevOps and wants to experience them quickly, it may want to consider turning to a service provider for help with either the technical or the “soft skills” challenges.

Some providers combine expertise in leading DevOps tools with business consulting, as well as proven best practices, frameworks and accelerators to assure the fastest and most significant ROI from DevOps. In some cases, we have seen such a mix of experi-

Organizations must implement end-to-end management and oversight to ensure that their slowest processes don’t delay deployment.

Quick Take

Sizing Up DevOps Best Practices

Based on our client experiences, we have identified the following DevOps best practices:

- **Maintain tight control over the scope and duration of each code update.** Limiting the size and complexity of iterations helps deliver the best balance of innovation and cost control.
- **Start small and build support for DevOps by finding a problem shared by both development and operations.** Using DevOps to solve a tightly defined problem, such as delays in moving code from development to production, can overcome mistrust and create support for a wider DevOps deployment.
- **Leverage developers or operations staff with DevOps skills by embedding them in each other’s organization, or in a cross-functional team to support a specific product or service.** This provides cross-training and an example of success and best practices for others.
- **Track and publicize success**, using metrics such as test cycle time, deploy rate and time, change lead time, defect rate in production, change failure rate and help desk ticket counts to show how DevOps is increasing quality and agility.
ence provide returns of up to 1,200% on the client’s DevOps investments.

Some providers can also help companies use the lean and continuous deployment enabled by DevOps to “incubate” new products, channels and business models. For example, our Experimental Learning Framework allows customers to build and test products quickly and at low risk. If a product or application does not meet expectations, it “fails quickly” and at low cost, delivering valuable insights to guide incremental and evolutionary improvement. Our innovative use of DevOps principles also allows organizations to quickly and inexpensively perform “A/B” testing to determine how different user interfaces and product offers affect conversion rates.

Footnote
1 2013 State of DevOps Report by Puppet Labs and IT Revolution Press.
About the Author

Kapil Apshankar leads Cognizant’s Technology Product Services Group, focusing on DevOps, Hadoop product development and next-generation product engineering. He brings a rich and diverse Silicon Valley perspective from spending a decade-plus exploring various technologies. In 2005, Kapil developed a sophisticated methodology that empowers companies to harness the power of multi-fidelity rapid prototyping, which fundamentally changes the way they build software products. In 2007, he helped build the first-of-its-kind enterprise experimental learning platform. In 2010, Kapil incubated one of the largest distributed Scrum development teams in India. He is a regular speaker at technical events, including O’Reilly Conferences and The Innovation Enterprise Summits. He can be reached at Kapil.Apshankar@cognizant.com.

About Cognizant

Cognizant (NASDAQ: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world's leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 164,300 employees as of June 30, 2013, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 500 and is ranked among the top performing and fastest growing companies in the world. Visit us online at www.cognizant.com or follow us on Twitter: Cognizant.