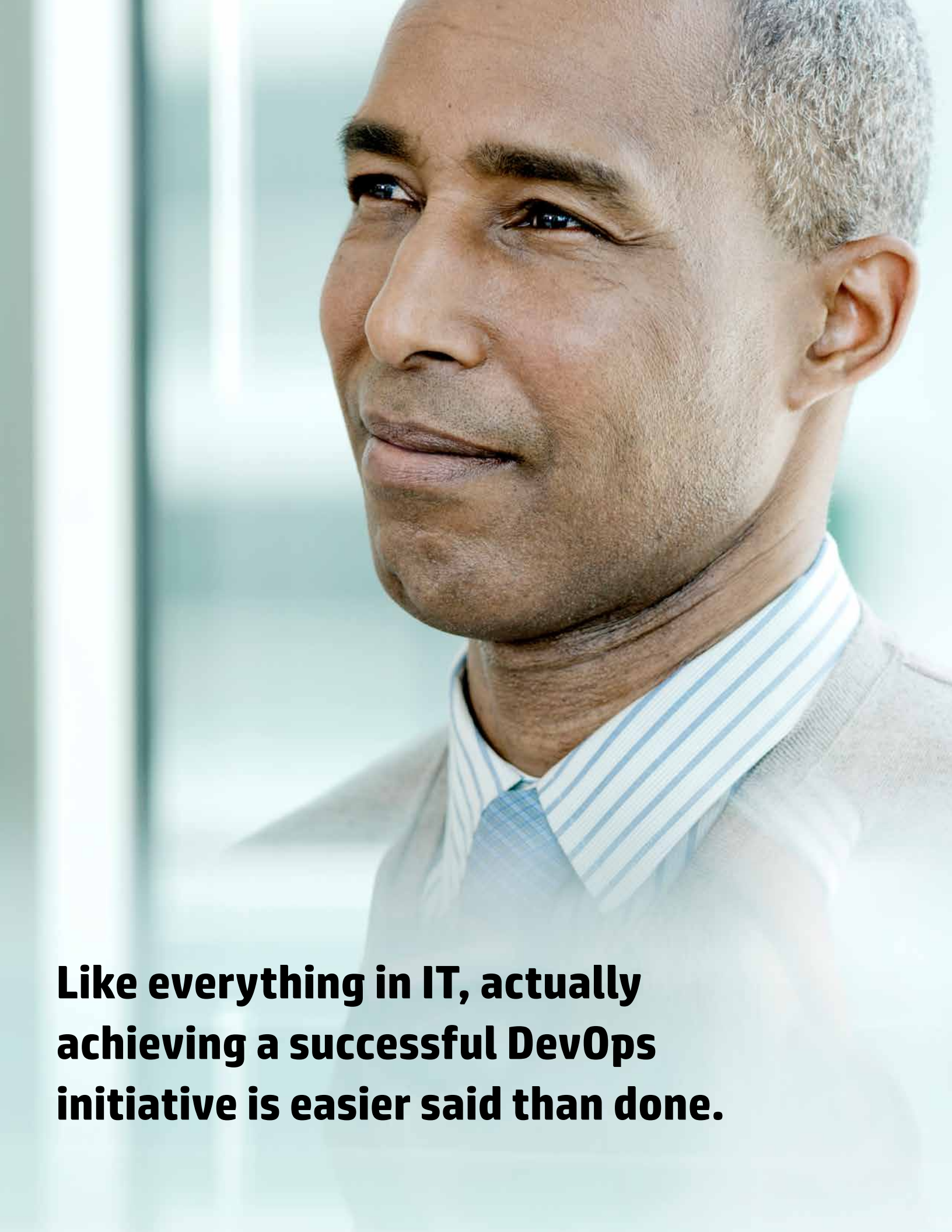


Business white paper

Target the disconnects between development and operations

Focusing on key challenges to
kick-start your DevOps journey





Like everything in IT, actually achieving a successful DevOps initiative is easier said than done.

According to Gartner, “Relations between Development and Operations are generally viewed as poor, with some even characterized as toxic.”¹ That’s pretty strong language. Hopefully, though, your situation is not quite as dire, and your development and operations teams at least talk to one another in the cafeteria.

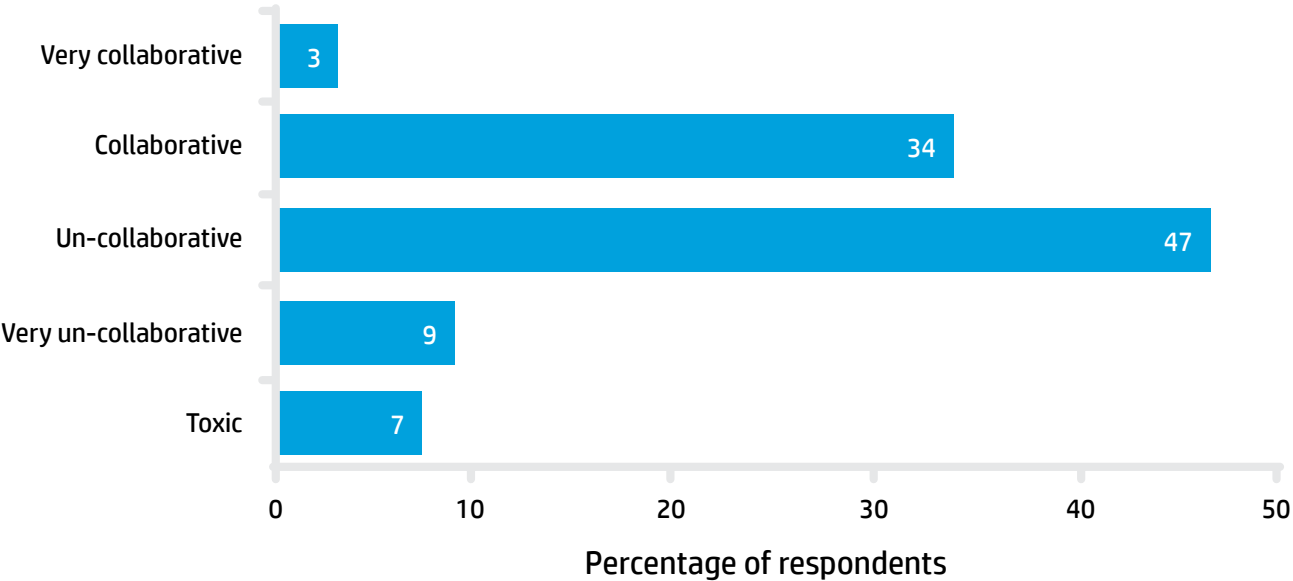
It is true, however, that in most IT organizations these two teams are separate entities, and the recent trend toward closer collaboration between them, commonly referred to as “DevOps,” indicates a general awareness on the part of IT that things would be better if application development teams and operations teams would work together more harmoniously.

But like everything in IT, actually achieving a successful DevOps initiative is easier said than done. Part of the challenge is that DevOps is still ill-defined—more a set of principles than targeted outcomes. We all agree that better collaboration would be a good

thing, but to what end? Being able to articulate your objective for undertaking a DevOps initiative is the first step. This will allow you to focus and prioritize your efforts so you can reap the greatest benefit. For many, the ultimate goal is end-to-end agility—shortening overall cycle-time from idea inception to delivering the value into the hands of your users. To achieve such an objective, IT leaders have to get down in the weeds and look at the specific disconnects that make the separation between teams a liability.

We developed this white paper to help you take a targeted approach to tackling DevOps. Based on extensive HP experience in helping IT organizations operate more effectively, we’ve identified seven common disconnects that you can translate into actions for unifying these traditionally siloed functions. By focusing on these specific challenges, you can turn the potentially toxic relationship Gartner mentions into effective collaboration that increases the overall velocity of your organization.

Figure 1
How would you describe the relations between your Application Development and IT Operations organizations?



1. The velocity mismatch

Possibly the most significant disconnect today between development and operations is that the two groups have very different perspectives on change and how quickly it should be introduced. This difference has been amplified in recent years by the emergence of Agile development methodologies aimed at what the Agile Manifesto calls “early and continuous delivery of valuable software.”

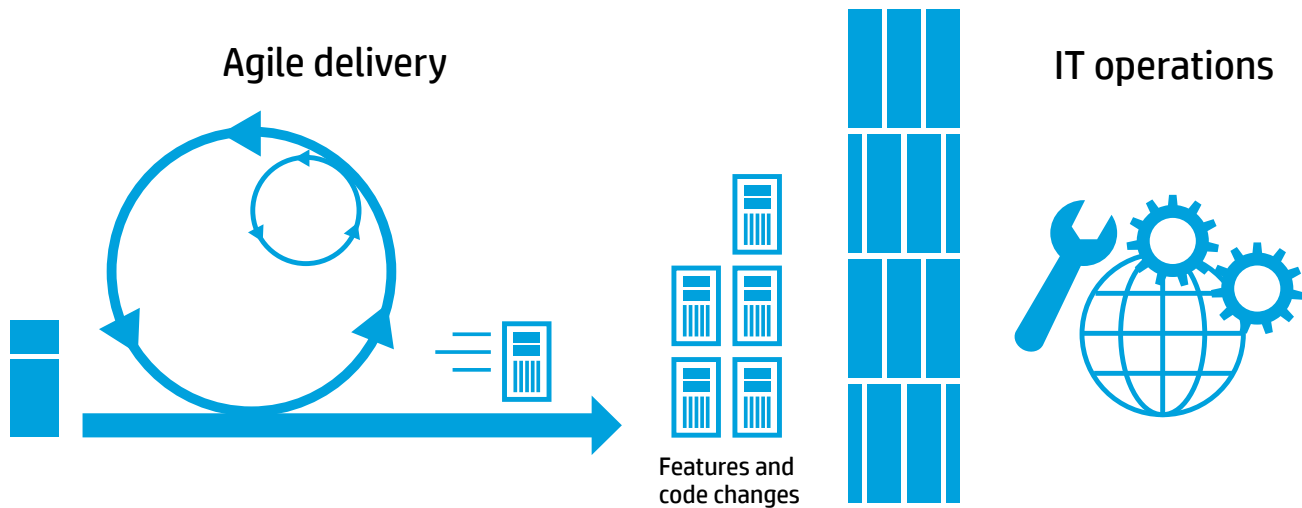
Development teams are working hand in hand with business stakeholders to rapidly deliver the change needed to take advantage of new opportunities and respond to competitive threats. In stark contrast, operations teams strive to tightly control change because they view it as introducing the risk of poor performance, outages, and security vulnerabilities. As development creates functionality faster, operations fears that the traditional testing practices they depend on to reduce those risks will be compromised for the sake of speed.

Reconciling these differing views is perhaps the biggest challenge to DevOps because it requires not just a process change, but also a cultural shift. To overcome this hurdle, the operations team must be reassured that lightening-fast changes are not jeopardizing its mission. Quality thus becomes the first and most critical bridge between the two groups. And the way to build this bridge is through a systematic approach to automated testing that begins at the point which change is introduced—code check-in by the developer. By automating testing and executing these tests as part of your continuous integration strategy, you can move toward a practice of near-continuous testing, dramatically shortening the feedback cycle to developers and allowing defects to be identified and fixed almost as soon as they are introduced.

A continuous testing regimen can incorporate a wide spectrum of testing: build, verification, regression, functional, performance, security, and acceptance. A rigorous, automated process that progressively verifies each dimension of quality will assure the operations team that velocity and quality can coexist.

Figure 2

Agile development speeds the creation of new features, but the advantage is lost if those features are bottlenecked in the release process.



2. Different measurements and incentives equal different behaviors

One straightforward yet fundamental change that can have a dramatic impact on how effectively development and operations work together is to reexamine how the two groups are measured. Having operated as silo organizations for decades has meant that most development and operations organizations have ingrained measurements that optimize their piece of the puzzle, but don't necessarily fit together well to achieve the end-to-end business objective.

Development teams have a long-standing tradition of being measured against the "iron triangle" of cost, time, and scope. The other dimension of quality is the one that, more often than not, gets short shrift (much to the operations team's dismay). Operations teams, as stewards of the business processes that keep revenue streams flowing, focus on and are measured by the stability and availability of the systems.

While these venerable approaches may make sense in isolation, they are myopic when it comes to the overall picture. And as long as the two groups are measured on very different things, their behavior will resemble competition more than cooperation. By aligning metrics, you can reduce friction and encourage the teams to pull together in the same direction.

Consider establishing aligned performance metrics across groups that take a holistic view. Examples include cycle time, percentage of releases that are successful on the first attempt, and percent of defects found in production.

And if at all possible, use a dashboard to create a single view across IT functions so that IT managers can set key performance indicators (KPIs) at the business level and cascade them downward to both development and operations—putting both teams on the same page with a shared world view and giving them mutual metrics and incentives.

3. An opaque wall between organizations

In addition to the alignment of measurements and incentives, development and operations teams will benefit from greater insight into each other's priorities, processes, and progress. Knowing what the other team is working on, how things are going, and when they'll be done eliminates surprises and confusion and contributes to overall efficiency.

Pulling down the organizational barriers and building DevOps collaboration can begin with something as simple as Agile teams inviting operations representatives to sprint planning sessions and end-of-sprint demos. Shared education sessions on each other's processes and pain-points can also provide a greater appreciation for each other's challenges and insight into how one team can make life easier for the other.

A common, integrated view into the state of development will aid in transparency and serve to connect IT functions. This allows all the stakeholders to openly see progress and things like quality, defect levels, and fix rates. Operations will be keenly interested in this, as they will ultimately be assuming responsibility for the applications. Collaboration and visibility throughout also mean a smoother handover when it is time for operations to take the changes.

In the other direction, operations should have tools that integrate back into the development environment. For instance, the support team should be able to easily channel production issues back to the development team so these can be quickly prioritized against other items in the backlog. This helps to make sure that development is working on the highest priority items for the business, irrespective of the item's origin.

Finally, to super-charge the idea sharing and problem solving that fuels day-to-day progress, larger or distributed teams benefit greatly from the latest in social-media-style collaboration tools that structure conversations around specific work items. This results in focused, context-aware discussions that bring the right people together to address a question or problem regardless of their team affiliation or location.

4. Functional versus nonfunctional requirements

Another important cultural difference is the fact that developers gravitate toward functional requirements, or what the business users want the applications to do, while operations is more concerned with nonfunctional requirements, or how the applications perform and behave once they are live.

If you ask a developer about the requirements for a new ATM feature, the response might be, “It has to access the history of each customer who uses our ATM; identify, based on the customer’s account balance and other factors, banking services the customer doesn’t have but might be interested in; and display a promotional screen on the ATM while the customer’s transaction is being processed.”

If you ask an operations engineer about the requirements for the same feature they might reply, “The service must execute within one fourth the time of the average ATM transaction, without exceeding available bandwidth, and support up to 10,000 customers accessing ATMs simultaneously.”

Trapped in its own definition of the requirements, development might deliver a beautiful and compelling splash screen that also happens to be such a bandwidth hog that it bogs down the ATM transaction. Or operations might insist that the application is working fine because it is executing within speed and bandwidth parameters, even though none of the customers are responding to the cross-sell promotions on the ATM screen.

Adopting the broader view is again the solution for resolving this disconnect. As seasoned developers have learned, the development mindset must expand to encompass not only functional requirements, but also non-functional requirements like performance and security. It means creating applications that are easy to enhance and that are easy to deploy, monitor, and troubleshoot. These things may not be as exciting from a development perspective, but they are absolutely critical from an operations perspective. Analysts actually estimate that for an application with a 15-year life span, more than 90% of the total cost of ownership comes after the initial build—in the running, maintaining, and enhancing of the application. This makes the build-to-run concept a true business imperative.

5. Different environments, different approaches

In most organizations, as a change is implemented it progresses through a series of environments such as development, test, staging, and production. Each of these environments serves a different purpose and a different set of stakeholders. They are typically managed in different ways, using different assets, often maintained by different people. The process of deploying to them typically consists of a raft of manual steps carried out by multiple people armed with lengthy checklists, which are often outdated or riddled with errors. This time-honored approach is far from agile and begs for mistakes to be made. In fact, it is a significant contributor to many of the headaches experienced between development and operations and the source of the constant refrain of “the build is broken” or “it works on my machine.”

This is an area ripe for automation. Automation allows teams to eliminate these manual handoffs, reduce errors, and accelerate overall release times. Fundamental to this is the notion of application portability, a capability that allows teams to seamlessly move applications from one environment to the next. Portability is achieved through environment-aware application models that are shared between development, test, and operations. Because everyone deploys the same way using the same assets, the result is consistent, accurate deployments every time across the various environments of the lifecycle.

6. Little to no sharing of assets

One of the most egregious sources of redundant effort is the “ownership” of assets. With their history as siloed organizations, development and operations teams create and then seem to hoard communications, resolutions, test scripts, usage information, and numerous other assets that could be shared and reused to eliminate duplication of effort and eliminate inconsistencies across the application lifecycle. In too many IT organizations, sharing doesn’t occur or is an afterthought—primarily because the potential benefits just aren’t recognized. Here are just a few examples where asset sharing and reuse can ease the burden:

- Automatically package test scripts created in development and send them to operations for production monitoring so operations need not recreate their own
- Import usage data from production to create more accurate and realistic testing scenarios
- Automatically convert real user sessions from production into performance scripts
- Leverage a shared knowledge-base of question, defect, and issue resolutions so you don’t reinvent the wheel for recurring problems

7. Work together, but remove dependencies

Closer collaboration between development and operations is a good thing, but sometimes DevOps can mean just getting out of each other's way. Reducing dependencies can help both teams work faster by removing handoffs and eliminating the need to wait for the other team to complete their tasks.

Start by identifying the key dependencies between the teams that result in latency or wasted time and work to remove or automate these pieces of the process. For example, allow development teams to manage their own environments using an automated lab management capability. Operations is involved with the initial set-up, but then developers and testers are free to provision and de-provision these environments themselves on demand. This provides speed and flexibility for development teams and frees operations up for other tasks.

HP and the three pillars of DevOps

The HP approach to DevOps rests on three solid pillars:

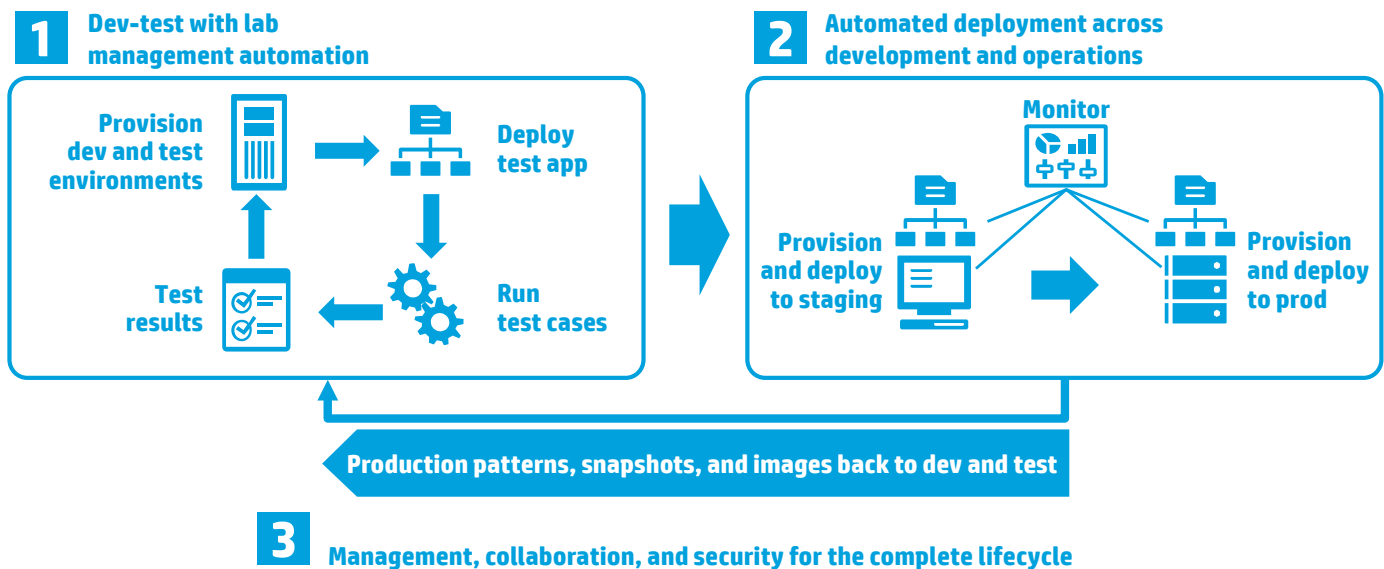
- **Quality:** the linchpin of the DevOps relationship and a prerequisite for rapid change
- **Automation:** to accelerate release times, eliminate manual handoffs, and reduce errors
- **Collaboration:** common goals, 360 degree visibility, and asset sharing to unify traditional IT silos

By applying these key elements, you can overcome the biggest obstacle in implementing DevOps—changing the traditional mindset of your teams. We support our approach with a comprehensive suite of integrated capabilities that can become the foundation for the integration of your teams.

¹ "Catalysts Signal the Growth of DevOps," Gartner survey, February 2012.

² Principles behind the Agile Manifesto.

Figure 3
The HP DevOps approach



HP can help you unify your development and operations functions to eliminate critical disconnects in traditional processes.

Get connected

hp.com/go/getconnected

Get the insider view on tech trends,
support alerts, and HP solutions.

© Copyright 2012 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

4AA4-2696ENW, Created August 2012

