TEAMQUEST

IT OPTIMIZATION INSIGHTS

DevOps Development: Keeping the Lights On

White Paper

BY LUIS COLON

TeamQuest specializes in IT Service Optimization

TEAMQUEST

Overview: The DevOps methodology embodies two core philosophies: decreasing the lead time of software deployment and the automation of delivery and testing. DevOps emerged as a practical response to the agile development movement, in contrast with traditional, phase-based or "waterfall" development, which is inefficient and labor-intensive. Traditional methods should be phased out, and companies should instead focus on continuous development and release. These newer methods accelerate performance improvements of software applications, reduce lead time, minimize resource utilization, and enable companies to predict service impairment. However, robust monitoring and analytics tools are necessary for DevOps to be successful, especially as cloud elasticity has complicated usage and cost tracking. With the right tools in place, DevOps has proved to eliminate waste, enabling IT teams to provide the highest service at the lowest cost.

Table of Contents:

I. Why DevOps?

DevOps is an exciting new philosophy being adopted by companies of all industries, but each vertical is drawn to it for different reasons. In IT, its main purpose is to decrease lead time: from application requests, to bug fixes, to product improvements, and to let users start using new capabilities sooner.

While DevOps provides a starting point for your strategy, how you accomplish this goal is up to you — lead time is most often reduced via a continuous delivery pipeline or by quickly deploying a canary release. In any case, the process should be automated to reduce waste from the development process, whether it be in the form of man-hours, over-provisioned resources, or unnecessary bugs. The goal, however, is not to present a flawless product, but rather craft an agile, robust process that can deliver workable solutions quickly and cost-effectively.

Through continual refinement, automated testing and deployment, and open communication, companies can revolutionize their IT maintenance, delivering the highest-quality product at the lowest possible cost.

While DevOps is growing increasingly popular, it has existed as a methodology for a relatively brief period of time. Less than eight years ago, at the Toronto Agile Conference, Patrick Debois famously introduced the topic of "Agile Development." The term was introduced as a possible resolution to the inherent conflicts between traditional development and operations teams through continuous communication, refinement, and feedback. That presentation sparked a larger discourse about combining those two disciplines into a single, coherent "DevOps" process.

Of course, DevOps comes as a response to the "agile development" movement, and adopts many similar methodologies: namely, the focus on rapidly improving customer-facing applications through close collaboration; forgoing traditional "waterfall" approaches to development, in which teams would operationalize products based on top-down design criteria, but rarely receive feedback or the chance to refine; and automating the testing and delivery of software using web-based tools.

By adopting those key aspects, companies gain an unprecedented speed and agility of development.

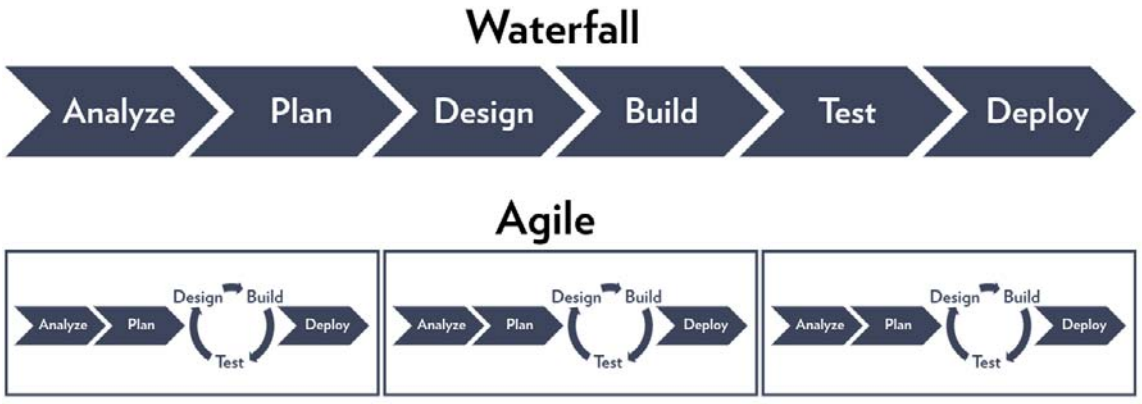
Other benefits of DevOps include:

- Improved application performance and IT system response times
- Fast and accurate responses to customer issues
- Flexibility needed to respond to market changes
- Alignment between customer needs and developmental action

Because DevOps links two previously distinct processes, it likewise doesn't require one tool, but a dynamic set of tools that some refer to as a toolchain. These tools typically fit into the categories: code, build, test, package, release, configure, and monitor, but it generally depends on the structure and goals of your company which of these categories ought to be emphasized or downplayed, but monitoring is always important.

Monitoring is Critical

Strong metrics and analytics are crucial to the testing phase of the development process. Without strong monitoring tools, development teams are essentially "flying blind" as they roll out new application changes — no analysis means no clue as to how changes will affect utilization of CPU, memory, disc I/O performance and ultimately, cost of supporting the applications. While new iterations may perform well in a test environment, things can be different once the application is



live. Waiting for service demand to spike or for a customer complaint puts undo pressure on your operations team to resolve the issue before costs or user problems create serious consequences for the business. To avoid these scenarios, proper predictive modeling is necessary to accurately test how the application may behave once it is live in your environment. Too often, companies focus almost entirely on testing the response time of their applications. An effective monitoring tool should be able to determine not just how well an application is responding, but how much IT resource it uses to do so, as well. Even if an application's response time is kept below a one-second threshold, for instance, it's not cost-effective if it uses an impractical portion of server resources.

However, by leveraging automation and analytics instead of error-prone manual development, DevOps processes can identify and rectify CPU and memory drainage, as well as other instances of overuse. With the right tools you can tell what it costs to deliver the response time you are seeing. Better yet, advanced predictive analytics can tell you what your fledgling release will cost in terms of resource utilization before you roll it out into production. You can balance functionality against SLAs and budget, making sure that IT will deliver the best possible return on the business's investment.

Accurately Predict with What-If Analysis

Spreadsheets and linear trending can lead to significantly overprovisioned forecasts. Queuing theory can give you more accurate prediction of future resource needs. Learn more at http://www.teamquest.com/what-if-analysis.

II. The Problem with Release Methods

Generally speaking, traditional release methods are only workable for new releases every six months or so, with in-between patches to fix devastating bugs. In the current IT marketplace, that's far too long to wait — if a team member finalizes a new feature in July for software that was released the month before, that update won't be available until the following January.

Moreover, without comprehensive regression testing, you're much less likely to notice a bug before the software is released, which means that brand reputation-damaging problems are inevitable. Even if you release a software patch to fix the problem, the problematic code is still embedded in your product — there's no way to really clear the drawing board once the product is in the hands of consumers.

The goal of DevOps is to reduce lead time, so that problems can be fixed quickly and that a solid brand reputation isn't nicked by software issues. With automated testing and monitoring, the likelihood of a bug reaching a customer (or a patch exacerbating the problem or creating a new one) is greatly reduced. And if development teams employ predictive analytics, they can anticipate application slowdowns before they occur. In this way, DevOps is can be proactive.

The Danger of Over-Provisioning

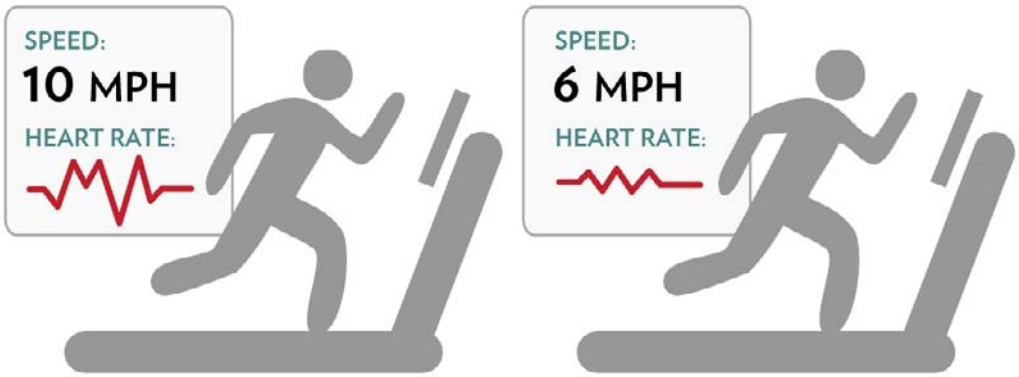
Many of the companies that already practice DevOps tend to place undue emphasis on application performance monitoring, often at the expense of resource utilization tracking. While there are plenty of available methods for monitoring and managing performance, the majority of them fail to measure server and memory usage in any meaningful way. As a result, applications will meet performance standards, but only because they consume massive amounts of resources in order to do so.

In essence, this is like powering a fast car by rapidly burning fuel; impressive to watch, but unsustainable and, from an engineering standpoint, unimpressive. This leads to a kind of over-provisioning that, without the right tools, becomes completely invisible. Without the tools to optimize system health, management of infrastructure cost becomes increasingly challenging. This explains why cloud cost management has become an increasing concern for ops teams over the last few years.

Incidentally, this consumer-first mindset maximizes neither performance nor cost-effectiveness, since hidden over-provisioning costs drain the capital resources needed for further refinement. This not only puts an upper limit on the possible performance of a given application, but also inflates the apparent performance of inefficient applications because of all the server resources propping it up. Companies need to correlate service demands with the size and costs of their infrastructure in order to optimize overall quality.

However, determining resource utilization and its costs isn't as simple as counting servers or purchasing new capital resources to meet demand — a largely linear project. Integration with the cloud has muddled how companies determine their usage and cost.

Performance vs. Health

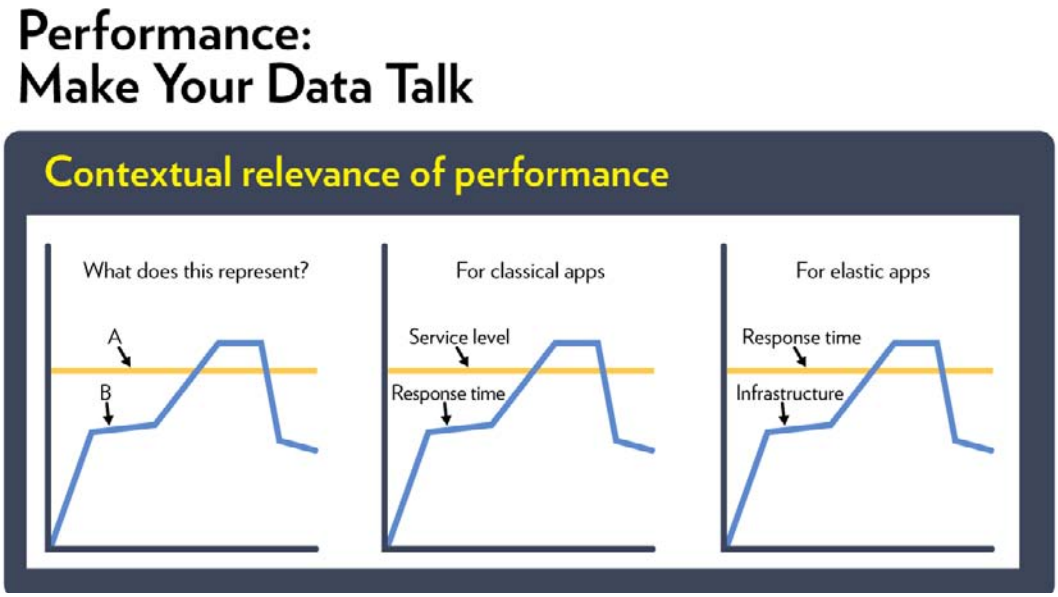


You can't judge how healthy your application is purely based on its performance. For instance, someone running ten miles an hour will be expending excess energy, and will eventually need more resources to do the same amount of work as a slower runner — without the right care, that runner might even collapse. Similarly, you want your application to give you the best possible performance, but not at the expense of your limited budget.

III. How the Cloud Complicates Usage and Cost

It used to be that the server costs of IT organizations were fixed, measured in a discrete number of capital expenditures. Today, many companies are opting for the flexibility of elastic, cloud-based applications. The assumption, of course, is that IT departments only pay for the capacity that they use. That isn't always the case, however.

The benefit of elastic applications and variable cloud clauses is that infrastructures automatically adjust to cope with rising demand so as to not violate SLAs. Unlike in traditional methodologies, where response time thresholds are breached for a short period when demand spikes, companies can guarantee that application transactions will never take longer than, say, two seconds. This is an obvious advantage, but the problem is that a company's choice of infrastructure capacity via various cloud server size options can have a massive impact on cost.



For example, a company might reserve a large cloud server instance for a worst-case demand scenario. But companies pay for every moment that that server is reserved, even if it's not being actively used to its capacity.

Of course, applications using large server instances or nodes can be dynamically moved into medium or small nodes for low-demand periods, which is relatively cheap. For heavy-duty use, however, that server must transition into a large node (say, 32 CPUs with 244 Gigs of RAM), which, if left running around the clock, can cost four to six times as much as would a physical server of the same size. Do companies need to reserve the large server? Or can smaller, less expensive instances handle demand with similar efficacy?

A separate, related problem is that traditional mainframe applications are not generally engineered to adapt to shifting cloud instances. For example, if demand dictates that cloud usage must be increased from two to eight clustered servers, an application must be immediately provisioned on six additional servers. Developed without a DevOps mentality in mind, those applications' availability is often incredibly inefficient at such moments, driving up costs even further.

Savvy IT departments must be able to creatively manage their cloud instance reservations to optimize for cost, either by turning off server usage at certain hours, or dynamically shifting between node sizes to adjust for elasticity. An IT department with only a performance monitoring tool in place would blindsided by these costs. To make effective server purchasing decisions and determine where your resource provisions are likely to fail, companies need usage monitoring and management. When automated, those processes will continually update project managers about weaknesses within their applications and infrastructure as they relate to both performance and cost.

IV. Why The Business Should Care About Revolutionizing IT Maintenance

The essential benefit of a DevOps mentality is that it turns the elimination of waste into a primary directive — and any waste removed directly translates into dollars saved. Now that improvements in technology and methodology are known

and demonstrable, waste comes in the form of any efficiency improvements that companies have been slow to adopt. These efficiency improvements can include:

- A more flexible infrastructural environment
- Highly efficient applications and server resource setups
- A reduction in man-hours spent performing non-automated testing and development

Today, an inability to change server utilization according to demand amounts to a needless underuse or overpayment of resources. In this way, executives can understand that revolutionizing IT maintenance and development is as simple as staying competitive.

However, it's easy to lose decision-makers in the technical jargon of DevOps — avoid putting cost-savings in terms of CPUs and resource utilization. It's just as accurate to say, "If we roll out Option A the IT resources necessary to meet SLAs will cost $1 Million. Option B, while including some highly desirable capabilities, will cost $2 Million."

Financial executives will also be glad to know that there are reasonable limits as to what you should pay for disaster scenarios. It's always best to plan ahead, of course, but continuously paying for headroom to weather a spike that would cost above six times the cost of your normal traffic isn't cost-effective. At that point, it's a better idea to ride out the spike than pay for a huge amount of capacity year-round, if dynamic elasticity is not yet an option.

Finally, a DevOps methodology ensures that the internal development of software and web-based applications is completely in-line with the customer's needs. Through continuous development, delivery, and testing, companies can deploy the leanest possible products with the highest possible impact. This saves costs for both you and the customer, while all but guaranteeing phenomenal service levels across the board.

V. DevOps Best Practice

With a limited budget, it's impossible to support a top-of-the-line product without sensitive calibration tools. In other words, you can't provide a Lexus with the budget of a Camry. To limit spending, many IT departments choose to either put an emphasis on optimizing the performance of its applications or reorient their server usage. Doing both at the same time often yields significant cost savings: since apparently healthy applications are oftentimes resource-inefficient, testing performance while keeping an eye on how resources are used is the best way of ensuring that you're actually performing efficiently.

Best Practices

While the application of DevOps as a discipline can seem fairly broad, there are a number a number of best practices that should be followed:

- Eliminate IT department silos in favor of system-wide thinking
- Maintain rigorous automated monitoring and analysis tools
- Ensure that tools have one-size-fits-all compatibility
- Monitor both performance and usage
- Automate processes wherever possible
- Gain complete visibility into IT infrastructure
- Leverage predictive analytics for proactive action
- Directly incorporate customer feedback
- Value process over product

There is no true limit to a DevOps methodology. Even if companies already monitor traffic, application response times, and transaction speeds, there are still other questions that need answering: how many transactions use CPU, where, and to what degree? How much memory does it demand? What about for network or disk space? How does it handle high concurrency spikes? When companies use tools to dive deeply into the available data, it quickly becomes clear where waste creeps into their processes. Without insight like this, DevOps can't be truly effective.

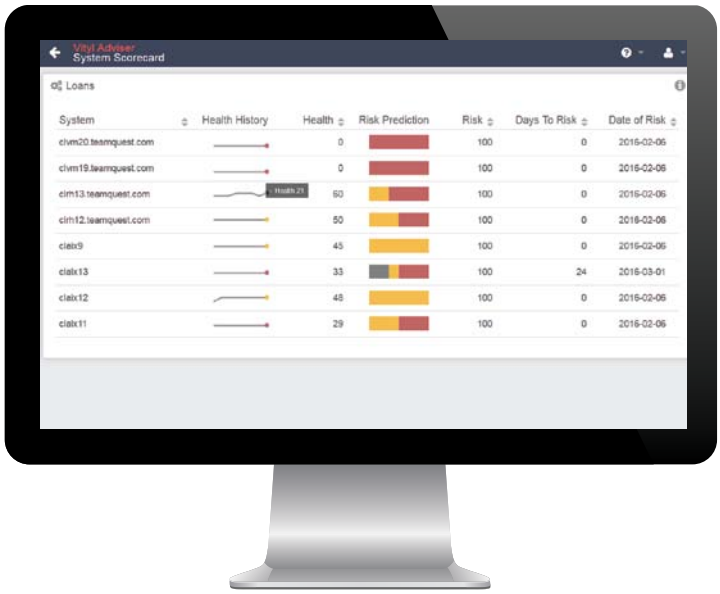
That's where TeamQuest's Vityl suite becomes an indispensable part of any DevOps initiative. Using best-in-class monitoring and analytics tools, companies gain comprehensive insight into performance and utilization, as well as correlations in-between. With greater accuracy than any other commercially available tool, Vityl allows IT departments to test the performance of new features before they launch, correlate traffic to resource utilization and cost, manage server buys and configurations, and continually refine DevOps processes themselves — all advantages that were not possible even a year ago.

In theory, DevOps offers an undeniable advantage in reducing the lead time of development and deployment over traditional models; TeamQuest empowers teams to make DevOps a practical reality.

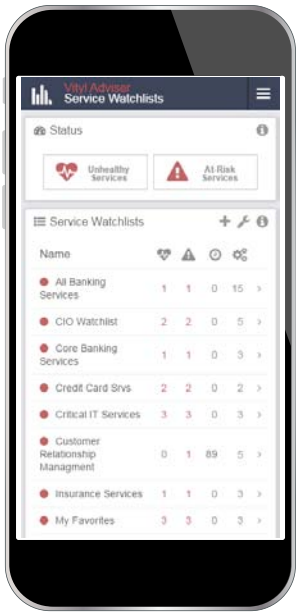
All of the methods adapt well to workload, configuration, and other environment changes. Using these methods with sophisticated algorithms, the final result is the most accurate health and risk calculations available in the industry, typically 95% accurate.

Vityl Adviser – Understand IT Health & Risk

Reduce your time to insight with automated predictive analytics



From Vityl Adviser's Service Health and Risk display you can see which servers in a service need attention. Further analysis can drill down to determine which components are past, current, or potential future bottlenecks.



Vityl Adviser lets you quickly understand the current health and risk of services in the context of the user. Watch lists are configurable by the user to encompass the relevant services.

Jerry Goldman, Director of Technical Services at Law School Admission Council, implemented Vityl Adviser from TeamQuest to streamline capacity management operations for his team.

> ***"Vityl Adviser's health and risk views give us quick insights across our entire IT environment. The Metric Drill Down views help us resolve problems and make decisions faster. The mobility and intuitive navigation is convenient and easy to use."***

To learn more about Vityl Adviser, visit **www.teamquest.com/vityl-adviser/**.

ABOUT THE AUTHOR - Luis Colon

Luis is a passionate and recognized leader in IT and agile development (XP, Scrum, Lean Startup, MAXOS) and frequent contributor to IT conferences and publications. He has multiple certifications in agile development including Scrum Master and Product Owner and SAFe Agilist. Apart from his role as Software Engineering Manager and Lead Architect at TeamQuest, Luis contributes content to TeamQuest publications from an engineer's perspective.

connect with TeamQuest:

TEAMQUEST

WORLDWIDE HEADQUARTERS

UNITED STATES

Executive Offices
TeamQuest Corporation
430 N 1st Ave
4th floor
Minneapolis, MN 55401

Development Lab
TeamQuest Corporation
One TeamQuest Way
Clear Lake, Iowa USA 50428

OTHER LOCATIONS

SWEDEN
GERMANY
UNITED KINGDOM
CANADA
MEXICO
HONG KONG

With resellers in many additional countries.

CONTACT US

info@teamquest.com
teamquest.com/about-us/contact-us/

DevOps Development: Keeping the Lights On