DevOps Without Dev is Dead on Arrival

August 2012

Introduction

Software applications are often the engine of every business today. To remain competitive, businesses need to innovate and respond to change fast without compromising quality.

To meet these needs, software development teams are increasingly adopting methodologies such as Agile to deliver software more frequently, and at the rapid pace demanded by the market. Building and testing these applications is one challenge, but getting them into production environments is quite another.

To achieve this faster pace and deliver high quality applications, DevOps has become an increasingly popular technique for shepherding software from the design phase through development and testing all the way into production. The business value of DevOps is quite profound: DevOps reduces software delivery times, improves application quality, and enhances the productivity of the development (Dev) and operations (Ops) teams. Most importantly, DevOps reduces the complexity, errors, and risk of application deployment failures in Production, the most important stage of the Application Delivery Pipeline.

In this paper, we explain what constitutes DevOps, describe what's driving the migration towards it, and why it's much smarter to approach DevOps from a 'Dev-first' attitude.

What is DevOps?

Let's understand exactly what constitutes DevOps.

First, Dev refers to the teams and procedures that create the software applications. The exact definition of Dev varies amongst organizations, but it typically incorporates:

- Software development
- Build, compile, and integrate
- Quality assurance
- Software release
- Support deployment and production tasks

Secondly, Ops refers to a wide swath of critical responsibilities such as:

- Provisioning and configuration
- System administration
- Health and performance monitoring of servers, key software infrastructure, and applications
- Change and release management
- Other infrastructure support

The tight collaboration and merging of principles, guidelines, and best practices between the Dev and Ops teams results in 'DevOps,' uniting principles and teams to provide a single holistic approach, bolstered by technologies such as release management and deployment tools.

A complete solution that extends from Dev to Ops makes it possible to answer critical application delivery questions such as:

- What is in this release?
- Who has tested it?

- What tests have passed and failed?
- Who approved the release?
- Who deployed the release?
- Can I recover easily from a failed release?

Thus, DevOps is a blend of processes, tools and cultures between the traditional Dev and Ops teams, intended to increase collaboration between the teams, reduce complexities and increase the pace at which the whole organization works together to deliver software applications.

Why is there a divide between Dev and Ops?

For decades, the Development and Operations organizations have been divided. This gulf exists for one key reason—a development team's mission and focus is to create value by producing change in response to customer and market requirements, while the operations team's mission and focus is to create value by keeping applications and services running reliably so that customers can depend upon them. Software applications and the delivery processes associated to deliver them to customers are complex - a higher change frequency directly correlates to a higher risk of failure.

Cultural differences regarding change

While it's dangerous to engage in broad stereotypes, it's fair to say that Dev tends to be more risk-taking while Ops is more risk-averse. After all, the Dev team is paid to create change and the Ops team is paid to keep the enterprise stable.

Through the adoption of various lean software development methodologies such as Agile, Dev has discovered that smaller and more frequent releases are better for producing higher quality software that is more likely to meet customer expectations and needs.

Meanwhile, experiences in software delivery have made Ops teams all too aware that the majority of downtime happens as an unintended side effect of making changes to software. In response, Ops has developed many mechanisms to guard the gates to software changes, such as heavy change management processes and encouraging fewer releases.

The technology divide: Process and tools

Driven by the desire for heightened business dexterity, many Dev organizations have either made the transition to Agile practices, or are seriously considering this type of conversion. The primary goals of Agile include smaller releases, more frequent delivery of working software, and a dramatically shorter time to market. In order to attain these ambitions, Dev tools have been growing more sophisticated, with heavy emphasis on boosting development and testing speed, improving collaboration among teams. The evolution of Dev tools enables software to be produced and delivered at an ever-increasing pace.

On the Ops side, Dev's augmented delivery pace has led to an increased demand for system resources. Ops tools have also grown in their sophistication to meet these needs. By automating system configuration, virtualizing entire software stacks, and enabling resources on demand via the cloud, Ops has transformed their tools and processes to assist the delivery of resources faster and more consistently.

Despite all the enhanced efficiencies supplied by modern Dev and Ops tools, it's still difficult to deploy software in a consistent, repeatable, and reliable manner. Dev often makes applications changes without involving Ops, resulting in Ops suffering Dev's decisions and holding the bag in customer-centric production environments when things go wrong.

Why is the migration to DevOps happening now?

Since the realities we just described have been in place for years, why is there such new found momentum towards integrated DevOps procedures now? The answer can be found in a series of business and technology-oriented fundamentals that have particular urgency today.

Business dynamics

Nearly every enterprise today faces extraordinary competitive pressures to deliver more, faster. This is a major cause driving the move to Agile practices. At the same time, no business can afford to speed up their software delivery process only to regularly implement unreliable, sluggish, buggy applications.

In the past, software was delivered relatively infrequently. This gave operations teams ample time to prepare and meet the inherent performance and stability mandates for enterprise-grade applications. Today, Agile practices have completely disrupted the traditionally leisurely pace of deploying software. Ops teams are now placed in the unenviable position of serving as a bottleneck to business agility.

Technology dynamics

Developing software today means juggling more platforms, teams, tools, and infrastructure than at any prior point in history. Meanwhile, Agile practices often translate into hundreds of delivery iterations per month.

In response, many Dev teams are investing in best practices and tools to support the software release process. This approach can provide highly desirable benefits, including:

- Holistic and coherent enterprise-wide processes
- End-to-end visibility
- Capturing and disseminating best practices
- Closed-loop analysis and reporting
- Consistent security policies

For their part, Ops teams are leveraging the power of virtualization and cloud computing to seek out new efficiencies. No longer is physical infrastructure holding up the delivery of applications. Instead, it's now up to the teams themselves to work together to quickly get applications to production.

In fact, the pace of innovation in both organizations means that mistakes are easier and faster to make and much harder to catch and track. With the growing need for audits and compliance, it is critical that Dev and Ops have complete visibility into each other's processes. Businesses now demand answers to challenging questions such as:

- What changes went into this application?
- Who approved these alterations?
- What packages comprise this application?

Without complete understanding of the entire Dev and Ops processes, it's not possible to answer these questions conclusively. To bridge this chasm, the Dev and Ops teams must collaborate much more than before. This collaboration needs to start at the Dev end of the cycle, where Ops can participate and learn along with Dev as early mistakes are made and corrected. This close coordination gives everyone ample opportunities to refine and fix any applications and their associated process errors as the application moves from Dev to Ops.

The Value of Dev in DevOps

For all of the reasons we've been describing so far, it's a good idea to start treating Dev and Ops as equally important participants in this process. This means embracing the Build-to-Ops cycle as a single, all-encompassing workflow, rather than continuing to treat it as several disparate components. It also requires aligning the previously incongruent organizational goals of both the Dev and Ops teams.

Three reasons to concentrate on Dev in a well-planned DevOps strategy

Many organizations interested in automating their software release and deployment processes focus exclusively on their "production" release/deployment processes—a task primarily managed by operations. In doing so, organizations don't fully leverage the critical value that Dev brings to DevOps.

The Dev team is the key stakeholder in creating, modeling and configuring applications. An implementation that's only focused on production release and deployment places a heavy burden of assumed application know-how on the Ops team. In reality, this group is much more conversant with systems and operations than application structures. This is usually the skill set they were selected for. A more productive approach is to fully leverage Dev application knowledge throughout the entire delivery process, starting from when Dev builds and deploys the applications in their own environments.

Dev also performs extensive application deployments in their environment (such as for quality assurance, user acceptance testing and pre-production staging). A production deployment-centric solution is unable to extract the deployment best practices that the Dev team creates throughout the delivery process prior to production.

For a successful DevOps implementation, it is recommended to start from a Dev-first perspective. There are three primary justifications for this approach:

1. Take advantage of Dev's comprehensive understanding of the application model

As the creator of applications, Dev teams are intimately familiar with what needs to be deployed. Dev can supply the Ops team with unparalleled insight into critical internal software attributes, such as:

- Application structure
- Required ancillary software and services
- Configuration settings
- Optimal application-specific runtime patterns

Understanding these characteristics is a prerequisite for a smooth software deployment process. By using a system that can leverage the Dev application "smarts," Ops can learn the application model and its infrastructure dependencies quickly and accurately.

2. Benefit from Dev's experience with the application deployment process

As part of their normal responsibilities, Dev teams conduct frequent application deployments into many development environments. These tasks repeatedly occur at many points in the Dev cycle, such as:

- Software development
- Integration testing
- Functional/system testing
- Acceptance testing

In response to these demands, Dev usually has a great deal of experience with multiple deployments. They need to, because they cannot fulfill their responsibilities without this knowledge. This expertise is very helpful when the time comes to place the solution into production. Examples of Dev's operational proficiencies include:

- Installing and adjusting platforms like operating systems and middleware to comply with application requirements
- Properly configuring the applications
- Fine tuning the applications to the specific functional need of the Dev tasks

Ops teams can leverage this expertise instead of starting from scratch each time they have to deploy an application iteration, each of which has its own inherent complexities.

3. Let Dev provide critical end-to-end visibility for minimizing potential deployment and production problems

Recall from earlier that a primary goal of the Ops team is to keep things running smoothly. However, with the frequent deployments mandated by Agile and other factors, their primary goal is at risk. Their failure can result in downtime, poor performance, errors, and a host of other undesirable outcomes.

As the owner of the application software, Dev teams have complete comprehension of what changes have been applied, their history, and who approved these updates. Pairing this proficiency with Ops' knowledge of their enterprise infrastructure yields complete end-to-end awareness.

A phased approach to achieving DevOps

Driven by business and technological trends, the move to DevOps continues to gain traction. To keep pace with this momentum, it's incumbent upon the development and operations teams to work together more closely. This requires cultural, process, tools, and automation changes to existing behavior.

The ultimate goal is to build a solution that enables 'one' team to own the application end-to-end throughout its lifecycle. This is best approached in a phased manner.

Phase 1: Share the goals

The first phase, where most enterprises find themselves today, is where both the Dev and Ops teams agree to work with each other and understand each other's focus and challenges. The common goal for both teams needs to be aligned around successful application releases. Ops has to understand and reconcile with Dev's goals and agree to stop being a roadblock to rapid application releases. At the same time, Dev has to understand the business goals that Ops is driven by and agree to work with the Ops team to share and alleviate their challenges.

At the end of this first phase, the enterprise can expect to reach a stage where infrastructure is available as a service (IaaS), self-service application deployments start to turn into a reality and both teams share 'pager duty,' i.e., share equal responsibility and work together to solve release deployment issues.

Phase 2: Share the requirements

After agreeing upon common goals, the teams need to move to the next phase, where both the teams have complete visibility into each others' requirements. Dev knows and uses Ops requirements and data feeds while designing and writing its applications and Ops has complete visibility into what's coming from Dev and how to release them starting within the Dev environments.

Very few enterprises find themselves in this phase, but they are getting there. The results of implementing this phase starts to bring significant rewards including transparent deployments and the ability to see the applications move through the different stages of the development lifecycle all the way into production (deployment pipelines). When a problem happens, the data connecting what broke and what was changed that made it break is available and shared so post-mortems can become more efficient and productive.

Phase 3: Share the processes

As the journey towards DevOps realization continues, the next phase should consist of the teams sharing their processes. As discussed in the paper, the best place to start is with the Dev organization. Dev creates the business applications and therefore, best understands what the application consists of, the pieces that need to come together, scripts and processes to successfully deliver the application and how to best orchestrate the moving pieces for the applications to deliver the best experience to end users. Ops can leverage this knowledge and experience as the applications move through the application development lifecycle to the final goal of delivering applications to users.

The ultimate goal of moving to DevOps is for both Dev and Ops teams to work as one team to achieve holistic end-to-end application management for all application releases. As a result, there should be no production deployment surprises, and real-time correlations of change to application operational incidents can be managed successfully. A very few, but emerging set of DevOps teams find themselves in this stage today. It is a tough journey, but for those who have made it, their ability to achieve faster time to market and improved quality control have been greatly enhanced.

Conclusion

The entire process from creating applications to delivering them is very difficult, with complexity inherently built into the applications themselves, the multiple environments that the applications are deployed to, numerous configurations and scripts built to deploy the applications and the sheer number of manual handoffs and processes involved between various teams from Dev to Ops.

To understand, simplify and manage these complexities, the DevOps journey must start from a Dev-centric approach as Dev teams best understand the applications and the processes around them. As the applications transition from Dev to Ops, learning from early failures and applying these lessons to refine the application and its associated deployment processes ensures that there are no production deployment surprises at the end when business results can be impacted.

About Electric Cloud

Electric Cloud delivers solutions that automate and accelerate the application development and delivery process. The company's award-winning products help development organizations to speed time-to-market, boost developer productivity, and improve software quality while leveraging the operational efficiencies provided by virtualized/cloud infrastructures. Leading companies across a variety of industries, including financial services, ISVs, mobile devices, semiconductors and transactional websites rely on Electric Cloud's automation solutions. For more information, visit **http://www.electric-cloud.com**.

Corporate Headquarters
Electric Cloud, Inc.
676 W. Maude Avenue, Sunnyvale, CA 94085
Tel: 408.419.4300 **Fax:** 408.419.4399
info@electric-cloud.com
www.electric-cloud.com

Electric Cloud Europe
1650 Arlington Business Park
Theale, Reading
Berkshire RG7 4SA United Kingdom
+44 (0) 1189 298280
europe.info@electric-cloud.com

Electric Cloud Asia Pacific
Suite 1905 Lippo Centre Tower 2
89 Queensway Admiralty
Hong Kong
Tel: +852.2918.8745
asia-info@electric-cloud.com

Electric Cloud Japan KK
22F Shibuya Mark City West
1-12-1 Dogenzaka, Shibuya-ku
Tokyo 150-0043 Japan
Tel: +81.3.4360.5375
japan-info@electric-cloud.com