DevOps for Windows

UpGuard™

INTRODUCTION

Once relegated to open source shops and dynamic, born-in-the-cloud software providers, DevOps is now widely adopted across organizations in a myriad of industries, from SaaS providers and hi-tech firms to financial services and banks. The methodology is becoming the de-facto way to manage software and IT services better and—unsurprisingly, since the vast majority of organizations are operating in Windows environments—DevOps for Windows has also come front and center as of late. If you're new to DevOps, this ebook can serve as a guide for your Windows-based organization's initial foray into DevOps. For others, it may offer insights into what a DevOps-enabled Windows shop looks and feels like.

THE PURPOSE OF DEVOPS

DevOps definitions abound, with the simplest being as follows: DevOps is IT operations merged with development. In the context of enterprises, this invariably means Windows-centric IT operations and development; as of April 2016, Microsoft held 89.23% of the OS market, a stark reminder of Redmond's ongoing dominance in the realm of personal/business computing. It's easy to overlook Microsoft's total market share when it comes to operating systems, especially with the groundswell of Mac and Linux users at the forefront of today's computing workforces. But as DevOps is vendor-agnostic, it's also not surprising that DevOps for Windows is really no different than DevOps at large. Of course, inevitable distinctions exist when it comes to DevOps tooling, some of which will be discussed later. But instead of defining the methodology per platform, a more relevant approach may be to define why organizations of all flavors need DevOps. Simply put, organizations need DevOps to deploy and manage software and IT services more simply, safely, and effectively.

THE RISE OF DEVOPS AND WINDOWS

Enterprise IT's role in the business has evolved drastically in the last decade. Now an integral part of the value chain, IT is no longer relegated to "plumbing" jobs for the firm's data pipes. Today's organizations see IT as an active player in enhancing and preserving all aspects of the business, especially when it comes to providing business continuity and delivering value to customers and end-users. This clearly extends IT's core values and functions beyond clicking the right buttons in Exchange, Sharepoint, or VMware's management console. For software development firms and SaaS providers, this means heightened communication, collaboration, and integration between software developers and IT operations professionals.

Earlier we stated that organizations adopt DevOps to deploy and manage software and IT services more simply, safely, and effectively. Specifically, the benefits of DevOps in this regard include:

- Increased frequency of deployments
- Increased collaboration between departments
- New software and services that would otherwise not be possible
- Reduced development and operations costs

DevOps represents a dramatic shift from traditional siloed development and script-heavy system administration. Tools that provide a common ground for developers and IT operations (i.e., a single pane of glass view)—as well as quality assurance, service management, security, and enterprise architecture teams—have therefore become commonplace in DevOps-enabled firms. And though open source languages and technologies like Python, Ruby, and Linux still dominate the DevOps tooling space, Windows-based tools are emerging to support .NET and Microsoft-centric environments. Additionally, an abundance of Windows-centric tools is now available for interacting with the Windows Server API: Powershell, WMI, System Center, DSC, TFS, Azure, to name a few.

DEVOPS AND THE WINDOWS GUI

Seasoned DevOps practitioners will often emphasize the importance of tooling in the context of shaping culture. Tools—as critical agents of change—are instrumental in both managing technology as well as shaping culture. It would therefore be disingenuous to say that DevOps, at least in practice per organization, takes the same form regardless of chosen platforms and vendors.

This is perceived by most as Microsoft's biggest Achilles' Heel when it comes to the DevOps playing field: the open source movement and community has traditionally been tied closely with Agile and DevOps, while Redmond has focused on providing proprietary software with sophisticated GUIs—a characteristic lacking of most open source projects. Indeed, even experienced Windows operators will attest to the flexibility and superiority of powershell commands over menus and icons.

At the end of the day, the tool best suited for the job should be used, regardless of whether or not it offers a slick visual user interface. It should address contemporary IT challenges in building/managing high-velocity organizations while facilitating constant improvement and collaboration between groups.

Because Microsoft has great GUI tools, they become the biggest strength and weakness of the DevOps/IT-Pro...

- Jeffrey Snover, Microsoft Technical Fellow and PowerShell Chief Architect

COLLABORATION AND CULTURE

Per DevOps, culture, collaboration, and communication are central to the way in which today's business gets done. To this end, tooling to support these three crucial pillars is essential; however, at the end of the day, nothing trumps bad culture. Buy-in from leadership is therefore critical to successful propagation of DevOps' shared set of values and behaviors across the organization, as any lasting cultural changes are implemented top-down.

Taking cues from the open source movement, DevOps espouses sharing, openness, and again—collaboration. Despite diverging from this model in the past, Microsoft has drastically changed its tune as of late: a new BSD Unix operating system, Windows 10 subsystem support for Ubuntu, and the recently open-sourcing of the Xamarin software development kit are a few examples of Redmond's commitment to open source. Tools help to create processes, which in turn influence behavior—as more Windows/Microsoft solutions move toward open source, you can certainly expect Windows-centric team cultures to follow suit.

VERSION CONTROL AND REPOSITORY TOOLS

Building a new software product (or modifying an existing one) requires the coordinated efforts of a development team. Large, highly complex applications (e.g., a 3-tier core banking app) may even include subgroups of developers, each working on a single module of the main application. To complicate things further, team members may be working in widely dispersed geographic locations.

Version control—also known as revision or source control—stores all the various software builds and versions created by each developer or group. Tools like Subversion (SVN), Git, Archiva, and Perforce are popular version control solutions. Some handle version control but do not contain a repository; instead, they are designed to integrate with specific repository tools. Maven is a common repository tool that works with several version control tools like Archiva, Nexus Sonatype and Artifactory. These are subsequently referred to as Maven repository tools.

Despite some variance in terms of features and capabilities, version control and repository tools share the following essential functions:

- **Version Tracking**: developers may wish to compare today's software version to yesterday or last year's. Version control systems keep track of every version for comparing performance between version and identifying/tracking bugs. Any problems triggered by a change can then be followed up with an examination of who made the change and the reasons for the change.

- **Trunks and Branching**: contemporary software developers are well-aware of the common tree motif: the application's main build is along a central line of development (i.e., the trunk); developers working on their own versions off the trunk (i.e., branches) must first test to ensure no conflicts or issues break the build.

- **Merging**: each branch's commit to the trunk is called a merge or commit. Merges are typically carried out by specific users called committers, but any developer can break off his/her own branch to work on it independently. Committers test them against the trunk's current version to ensure branches do not conflict with or break the trunk's stability. Upon merge test failures, the responsible developers are informed so they can make the necessary changes.

Version Control and Repository Tools:
Artifactory, Archiva, BitBucket, GitHub, Perforce, Nexus, Team Foundation Server

BUILD TOOLS

Build tools are software products designed to automate various software development tasks, the most important of which is arguably compilation of source code into binary code. Specifically, they automate the following::

- Packaging the binary code
- Running tests and test cases on the code
- Deploying programs to production systems
- Sending notifications of job completion
- Creating documentation and release notes

A recent capability of modern build tools is distributed processing, which allows for each step in a process or workflow to be sent to a different node for execution. For example, a post-step of the build may require the execution of multiple test scripts on multiple machines. In this case, a distributed-processing build tool can send the different test scripts to different machines. It's worth noting that distributed processing is different from distributed builds, which entails breaking up the seemingly monolithic compile/link process into different steps and sending them—dependencies and all—to different machines for execution and recombining the sub-steps into one whole. This requires a sophisticated level of machine intelligence; as such, not all tools can perform distributed builds.

Build Tools:
Bamboo, Ant, Maven, Perforce, TeamCity

CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY

Continuous integration and continuous delivery (CI/CD) refers to the software development practice in which developers regularly merge their self-testing code to a shared central repository several times a day. Each merge is then verified by an automated build tool that enables teams to detect problems early. By splitting up the work into smaller, pre-tested units, integration errors are greatly minimized, freeing up more time for actually building and improving the product.

Some common CI/CD best practices include:

- Developers never break the build. Code should be thoroughly tested and built before integration into the central repository. The test build should be done on a clone of the production environment to prevent breaking the main build with the excuse of "but it ran fine on my computer."

- Members of the development team start each working day by updating their project or branch from the central repository.

- Developers must integrate early and often. It is up to each individual to split their work into logical units that yield atomic commits. That is, each commit must be for a single, logical unit of work.

- Developers commit to a baseline every day. Whenever a commit happens, a central CI server builds the system and runs unit and integration tests.

CI/CD processes use their own specialized tools, some of which are noted below. They are also closely linked to build automation tools like Maven and the Visual Studio suite.

CI/CD tools:
Bamboo, Hudson, Jenkins, CruiseControl, TeamCity.

CONFIGURATION MANAGEMENT AND AUTOMATION

Configuration management (CM) and automation are the collective processes and tasks involved in rolling out, tracking, and controlling changes to software, apps, and hardware, as well as the detailed recording of an organization's IT assets. CM primarily consists of two key components:

1. Maintaining an up-to-date inventory
2. Management of changes to that inventory

It's worth noting that in contrast to version/revision control, CM does not generally consist of changes to software source code.

Most modern CM and automation tools generally use agents installed on all nodes to monitor and implement changes. These agents can either be controlled by a master, or in a few cases, are configured in a peer-to-peer setup with no central, controlling "brain."

The following are a few leading CM tools for Windows environments. These setups typically consist of a Windows domain controller, an Active Directory setup, 2 or more Windows servers running production/test application systems, and an environment of Windows-user client nodes.

CM/automation tools:
Chef, Puppet, Ansible, Puppet, UpGuard, RunDeck, SaltStack, SCCM

TEST AUTOMATION

Test automation is the use of specialized tools to execute tests in a controlled and repeatable manner. As opposed to manual human testing, test automation allows for reliable, repeatable results as well as unit testing and load testing, with some tools even offering the ability to test GUIs by mapping and manipulating interface elements—buttons, URLs, text boxes, and more.

The ability to organize and list bugs or issues uncovered during testing—known as issue tracking— is often paired with test automation. Issues are numbered or labeled, assigned to specific people in the development and/or testing team, and tracked for progressive resolution and final closure.

Test Automation Tools:
Jira, Bugzilla, TestComplete, QTP, Cucumber

CONTAINERIZATION

Containers are all the rage as of late, and for good reason: as lightweight alternatives to full machine virtualization, they encapsulate applications in containers with their own operating environments. This effectively does away with dependency and configuration issues by providing a consistent environment for software to run in, from development through production.

Docker is perhaps the most well-known of the containerization lot, but a handful of others exist. Microsoft will support Docker containers in its Azure cloud platform but is developing its own container technology called Drawbridge.

Containerization Tools:
Docker, Rocket, LxD, Turbo.net.

CONTINUOUS SECURITY AND COMPLIANCE

The importance of continually testing and monitoring one's infrastructure for vulnerabilities, configuration changes, and drift cannot be stressed enough. Developers may be savvy enough to avoid code-level security issues in an application, but ultimately the software is as vulnerable as its underlying systems and infrastructure. Detecting and remediating security flaws at all levels of the application and technology stack is therefore crucial to bolstering a software application against security threats and potential compromise. Implemented as part of the continuous integration process in ongoing software iterations, continuous security testing and monitoring help to maintain a strong security posture throughout all phases of development.

For organizations operating in highly regulated industries, DevOps can be a key enabler for ongoing compliance. Because the methodology espouses principles like process repeatability and continuous testing, firms can use its tools and practices to ensure that environments and IT assets are in compliance on an ongoing basis. Platforms like UpGuard provide automatic, policy-driven testing and monitoring to ensure that your Windows-based infrastructure is in line with regulatory standards.

Continuous Security/Compliance Tools:
UpGuard, Chef Compliance, Amazon Inspector

VISIBILITY AS A PRECURSOR TO DEVOPS

The cross-functional collaboration and information un-siloing promoted by DevOps is deceptively straightforward in theory, but challenging in practice. Much of this is due to the sheer volume of disparate moving parts required to make the DevOps machinery operate: developers checking in/out and merging application code, operators standing up/bringing down/patching systems, and any number of other processes for delivering software and IT services. Suffice to say, it's difficult to optimize what you don't understand or completely have a handle on.

For this reason, visibility is arguably the chief precursor to DevOps. It's been said that you can't automate what you don't understand; this, along with the natural tendency for system configurations to drift over time, make a single system of record crucial for enabling smooth DevOps adoption and ongoing optimization. Having a single system of record in place enables proper visibility and validation for consistent delivery of quality software and IT services. To this end, UpGuard closes the feedback loop to ensure that developers begin from the same state as production, post-automation states are in line with expectations, and infrastructures are monitored against an up-to-date, secure "golden images."

MOVING TOWARDS ENTERPRISE RESILIENCE

Chalk it up to being a victim of its own success, but the reality is that Windows on an aggregate scale is more prone to being compromised. This is because the popular operating system finds itself in the crosshairs of attackers more often, usually in the form of personal desktops, enterprise database servers, PoS devices, and a myriad of other high value targets. But Windows isn't approaching its end-of-life any time soon and neither are cyber attackers.

The solution to not just surviving—but thriving in today's digitized economies—is digital resilience, or the managing of cyber threats as a function of enterprise risk management. Since certain digital risks must be taken to remain competitive, organizations must design their businesses to withstand IT security failures and data breaches, as opposed to treating security as a binary state. This is akin to controlling crime in big cities: attempting to thwart every instance of crime is both unfeasible and inefficient; instead—by instituting selective controls and effective monitoring/response—one can address the majority of criminal activity inside a municipality.

However, digital resilience is not just about maintaining strong IT security. It's about effectively managing all of the risks associated with doing business in today's digital landscapes. This includes continuous compliance to constantly changing regulatory laws as well as ongoing integrity in rapidly expanding, increasingly disparate environments. Perhaps unsurprisingly, many organizations that have adopted DevOps practices have found the road to digital resilience easier to traverse. Those familiar with the methodology and its espousing of collaborative efforts will recognize parallels to resilience.

For example, DevOps is by nature people-centric; similarly, resilience promotes information security measures that emphasize individual accountability and trust. Security and compliance must be treated as organization-wide concerns, not just relegated to IT or security groups. Since all parts of the business these days transact digitally—often outside of the reach of corporate IT—strong security must be an initiative shared by all employees. Additionally, to survive in today's cyber threat landscape, a multi-layered approach is necessary that combines traditional defenses like firewalls and endpoint security with new advances such as SIEM, crowdsourced intelligence, and integrity monitoring/validation, among others. However, these mechanisms must fall against the backdrop of an enterprise's overall resilience strategy. Security controls should be prioritized and layered based on the company's constraints, keeping in mind that the speed of change in today's environments make it impossible to anticipate and counter every cyber attack.

CONCLUSION

Today's always-on consumer mentality is reshaping both the customer's expectations of the enterprise as well as IT's role in the business at large. DevOps is the primary enabler of these changes, and in a world dominated by Windows-centric software, you can be sure that Microsoft is not resting on its laurels when it comes to DevOps initiatives. For example, Visual Studio Team Services—formerly known as Visual Studio Online—consists of a set of services aimed squarely at DevOps practitioners. Any team, any language, any tool—this is the Visual Studio Team Services mantra.

Still dubious about Microsoft's commitment to DevOps and open source? Check out the release of Visual Studio's source code to Git. And for more on DevOps for Windows, check out Microsoft's Technet blog resources dedicated to all things DevOps.

It's about absorbing the punches and bouncing back from the big things while accepting certain risks for the achievement of success...

- Peter Firstbrook,
Research Director at Gartner

SOURCES

http://www.extremetech.com/computing/227693-windows-drops-below-90-market-share-for-the-first-time-in-years-windows-7-falls-below-50

https://www.microsoft.com/en-us/download/confirmation.aspx?id=46920

http://www.webopedia.com/TERM/C/containerization.html

http://www.serverwatch.com/server-trends/docker-not-the-only-container-option-in-2015.html

http://www.ubuntu.com/cloud/lxd

http://www.zdnet.com/article/why-microsoft-is-turning-into-an-open-source-company/

http://research.microsoft.com/en-us/projects/drawbridge/

https://github.com/Microsoft

http://blogs.technet.com/b/devops/

http://www.extremetech.com/computing/227693-windows-drops-below-90-market-share-for-the-first-time-in-years-windows-7-falls-below-50

http://betanews.com/2016/02/05/windows-10-hits-18-percent-usage-share-in-the-enterprise/

UpGuard™

Taking your business through a digital transformation depends on trust. UpGuard creates trust within and between businesses by providing unparalleled visibility into computing environments and translating the resulting IT security posture into business risk in terms non-technicians care about.

Trust your IT environment again.

UpGuard™