
Es
DevOps:
La Perspectiva de
Operaciones
Don Jones
Principal Author
THE
DEVOPS
COLLECTIVE
devopscollective.org
PowerShell.org

DevOps: The Ops Perspective (Spanish)

The DevOps Collective, Inc.

Este libro está a la venta en
http://leanpub.com/devops-the-ops-perspective-spanish

Esta versión se publicó en 2018-10-28

Este es un libro de Leanpub. Leanpub anima a los autores y publicadoras con el proceso de publicación. Lean Publishing es el acto de publicar un libro en progreso usando herramientas sencillas y muchas iteraciones para obtener feedback del lector hasta conseguir tener el libro adecuado.

También por The DevOps Collective, Inc.

Creating HTML Reports in Windows PowerShell

A Unix Person's Guide to PowerShell

The Big Book of PowerShell Error Handling

DevOps: The Ops Perspective

Ditch Excel: Making Historical and Trend Reports in PowerShell

Secrets of PowerShell Remoting

The Big Book of PowerShell Gotchas

The Monad Manifesto, Annotated

Why PowerShell?

Windows PowerShell Networking Guide

The PowerShell + DevOps Global Summit Manual for Summiteers

Why PowerShell? (Spanish)

Secrets of PowerShell Remoting (Spanish)

The Monad Manifesto: Annotated (Spanish)

Creating HTML Reports in PowerShell (Spanish)

The Big Book of PowerShell Gotchas (Spanish)

The Big Book of PowerShell Error Handling (Spanish)

DevOps: WTF?

PowerShell.org: History of a Community

Índice general

DevOps: La Perspectiva de Operaciones

Por Don Jones

"DevOps" es un término muy popular en estos días - pero ¿qué significa realmente para una persona Ops? Este libro de alto nivel intenta poner a DevOps en perspectiva con ejemplos y descripciones del mundo real.

Descargar el código El módulo EnhancedHTML2 mencionado en este libro puede encontrarse en PowerShell Gallery[1]. Esa página incluye instrucciones de descarga. PowerShellGet es necesario, y se puede obtener de PowerShellGallery.com

¿Ha sido útil este libro? El (los) autor (es) le pide (n) que haga una donación deducible de impuestos (en los EE.UU., consulte sus leyes si vive en otro lugar) de cualquier cantidad a The DevOps Collective[2] para apoyar su trabajo.

[1] https://www.powershellgallery.com/packages/EnhancedHTML2/
[2] https://devopscollective.org/donate/

Revise las actualizaciones! Nuestros ebooks se actualizan a menudo con contenido nuevo y corregido. Los hacemos disponibles de tres maneras::

- Nuestra rama principal GitHub organization[3], con un repositorio para cada libro. Visite https://github.com/devops-collective-inc/
- Nuestra GitBook page[4], donde puede navegar por los libros en línea, o descargarlos en formato PDF, EPUB o MOBI. Utilizando el lector en línea, puede saltar a capítulos específicos. Visite https://www.gitbook.com/@devopscollective
- En LeanPub[5], donde se pueden descargar como PDF, EPUB, o MOBI (login requerido), y "comprar" los libros haciendo una donación a DevOps. También puede elegir recibir notificaciones de actualizaciones. Visite https://leanpub.com/u/devopscollective

GitBook y LeanPub generan la salida del formato PDF ligeramente diferente, por lo que puede elegir el que prefiera. LeanPub también le puede notificar cada vez que liberamos alguna actualización. Nuestro repositorio de GitHub es el principal; los repositorios en otros sitios suelen ser sólo espejos utilizados para el proceso de publicación. GitBook normalmente contendrá nuestra última versión, incluyendo algunos bits no terminados; LeanPub siempre contiene la más reciente "publicación liberada" de cualquier libro.

[3] https://github.com/devops-collective-inc
[4] https://www.gitbook.com/@devopscollective
[5] https://leanpub.com/u/devopscollective

¿Qué es DevOps?

"DevOps", como un término, no tiene una definición realmente concreta. Es una filosofía, una forma de trabajar, y significa cosas diferentes para diferentes personas. En su mayor parte, la comunidad DevOps generalmente acepta la definición del artículo de DevOps en Wikipedia, que en parte dice:

> … un método de desarrollo de software que hace hincapié en la comunicación, la colaboración, la integración, la automatización y la medición de la cooperación entre los desarrolladores de software y otros profesionales de las tecnologías de la información (TI).

Personalmente no siento que la definición sea mala. DevOps es mucho más que un "método de desarrollo de software". Sin embargo, vamos a jugar por un momento, y a reconocer que el software gobierna el mundo. El presidente de los Estados Unidos no se puso de pie y dijo, "todos deben aprender de redes", él dijo, "todo el mundo debe aprender a programar". Internet es enorme, es un conjunto de ingeniería impresionante, de infraestructuras de redes como nunca se había visto, pero es en su mayor parte una tubería "tonta" utilizada para entregar software. Software es de lo que se trata. Pero el software no llega a ninguna parte solo, ni hace nada, sin una infraestructura que lo soporte. Ambos (software e infraestructura) trabajan juntos, y son lo que hace que la tecnología sea útil. Es por eso que DevOps comprende tanto "Desarrollo" *como* "Operaciones". Así que, para este libro, me gustaría tomarme la libertad de volver a definir ligeramente a DevOps como:

> … un enfoque de la gestión de la tecnología que hace hincapié en la comunicación, la colaboración, la inte-

> gración, la automatización y la medición de la cooperación entre los desarrolladores de software y el personal de operaciones de TI con el fin de crear y entregar aplicaciones de software a sus usuarios.

Es importante comprender que DevOps es *una cosa tan grande* que es casi imposible verlo todo a la vez. Se trata de numerosas técnicas, múltiples funciones dentro de una organización (es por eso que la "cooperación" está en la descripción, junto con "colaboración"), y un montón de tecnologías que se cruzan. Pero no se preocupe. Este libro no va a tratar de cubrir todos esos aspectos..

Algunos Antecedentes

El término "DevOps" probablemente fue acuñado por Patrick Dubois[6], inspirado en una presentación de Velocity en 2009 de John Allspaw[7]. Filosóficamente, está inspirado en gran parte por las enseñanzas de "Lean Manufacturing" de luminarias como W. E. Deming, Taiichi Ono, Eli Goldratt y otros. Eso es importante, porque esos señores basaron sus pensamientos en la premisa de que la mayoría de *los trabajadores quieren hacer un buen trabajo*. Un hilo común en Lean Manufacturing - y de hecho un punto específico del enfoque de Deming - era poner fin a la confianza en "QA" como un medio para lograr la calidad. Sí, usted establece las medidas adecuadas para ayudar a la gente a prevenir sus propios errores, pero olvida poner las "puertas", resultará asumiendo que sus trabajadores son maliciosos o incompetentes. Ese principio a menudo se convierte en el mayor obstáculo en la adopción de Lean Manufacturing, DevOps, o cualquier otra cosa que derive de ese principio.

[6] http://jedi.be/blog
[7] https://www.youtube.com/watch?v=LdOe18KhtT4

DevOps, para Ops

En su lugar, este libro examinará el microcosmos de DevOps relacionado más específicamente con la parte Ops. En cualquier organización que intente implementar un enfoque de DevOps, el lado operativo de la casa necesita de ciertas capacidades. En muchos casos, la parte operativa de la organización debe proporcionar un nivel de automatización y una especie de autoservicio, que le permita al departamento de desarrollo abstraerse de toda intervención operativa. Las operaciones a este respecto tienen que ver con la implementación de métodos seguros, manejables y monitorizables que permitan la liberación de software de manera rápida, sin que los proyectos se conviertan en una carga operativa importante. Exactamente cómo se procede, y qué capacidades se proporcionan, variará grandemente dependiendo de cada organización.

Para proporcionar esas capacidades, Operaciones tendrá que promover una cierta cantidad de desarrollo de software, de tal manera que se creen unidades de automatización que hagan que el lado de Operaciones de la organización funcione de manera más independiente. Esos esfuerzos de desarrollo de software pueden llevarse a cabo de una manera muy centrada en DevOps, y este libro se centrará en gran medida en esa actividad.

Es una Filosofía

DevOps se parece a la contabilidad, en que es un conjunto de principios abstractos, enfoques y patrones. En la contabilidad, se tienen prácticas contables generalmente aceptadas, o GAAP. No son reglas, per se, pero son tan generalmente aceptadas que llevan el peso de la ley de muchas maneras. DevOps es o puede llegar a ser así, ya que puede incorporar un conjunto de prácticas y enfoques que generalmente se reconocen como el mejor camino a seguir. Contabilidad también tiene herramientas que le ayudan a

implementar sus prácticas. QuickBooks, por ejemplo, es un paquete de software que encarna y hace cumplir una gran cantidad de prácticas contables, por lo que es más fácil ponerlas en práctica en su organización. Del mismo modo, el mundo de DevOps tiene una serie de herramientas - muchas aún por nacer por lo que DevOps es relativamente nuevo - que le ayudan a implementar prácticas y enfoques de DevOps. A medida que el mundo de DevOps intenta cosas, aprende de ellas y perfecciona sus enfoques, así que podrá encontrar más y más herramientas creadas para ayudar a hacer esos enfoques más fáciles y más consistentes de aplicar en el mundo real. En este libro, nos centraremos mucho más en las prácticas y patrones que en las herramientas, de modo que podamos permanecer en un nivel superior y no forzarle a comprometerse con una pila de tecnología en particular.

A diferencia de la contabilidad, y como ya he mencionado, DevOps es relativamente nuevo. Y, a diferencia de la contabilidad, DevOps vive en un campo que está en constante evolución y cambio. Así que no espere cosas como, "aquí está lo que debe hacer" o reglas y regulaciones. En su lugar, la práctica de DevOps es actualmente el 80% de la teoría, el 10% lo que la gente ha experimentado hasta el momento, y el 10% pura conjetura. Hay un montón de empresas experimentando con los enfoques de DevOps, por lo que es una industria todavía estamos descubriendo. Gran parte de este libro se centrará en lo que se ha hecho con éxito en otros lugares, y buscará concretamente lo que las Operaciones ofrecen en esas situaciones.

Es un enfoque

Entender sobre todo que DevOps es un enfoque de gestión de la tecnología. Sugiere maneras de gestionar proyectos, maneras de gestionar el desarrollo de software y maneras de gestionar las operaciones. Dicho esto, sin la administración de buy-in en su organización, no se puede hacer DevOps. Así que, si usted

está pensando, "*bueno, en mi organización nunca vamos a apoyar la idea de que los desarrolladores "empujen" el código directo a producción*", entonces puede dejar de leer ahora mismo, a menos que esté interesado por curiosidad. Este libro, al menos, no va a intentar convencerlo de las ventajas de DevOps – eso ya se ha hecho en otros libros. Este libro supone que ya ha aceptado los beneficios de DevOps y que está interesado en profundizar un poco más en lo que eso significa para un equipo de operaciones de TI tradicional..

No hay tal cosa como un equipo de DevOps

Y seamos muy, muy claros: usted no puede tener un "Equipo de DevOps" en su organización. Eso es absurdo. *DevOps es un enfoque de gestión* que abarca el desarrollo de software, administradores y operaciones como una sola unidad. Todos trabajan juntos para facilitar la creación y el despliegue de aplicaciones. Es posible que sólo uno de los muchos proyectos internos se llevará a cabo en una forma DevOps - pero dado el tipo de cambios que Ops tendrá que hacer para facilitar el enfoque de DevOps, va a ser difícil de "hacer" DevOps de forma fragmentada. Sólo tenga en cuenta que - DevOps trata sobre cómo cambiar la forma de hacer negocios Si no se siente a gusto con esta idea, entonces siempre va a sentir algo de miedo

Puede tener equipos o proyectos específicos dentro de su organización que actúen de una manera DevOps, siempre y cuando el equipo sea lo suficientemente funcional para proporcionar todas las disciplinas de Dev, Test, Ops, etc., que sean necesarias. Así que la totalidad de la propiedad de TI no necesita "aplicar DevOps", pero un proyecto individual si podría. Dicho esto, tener sólo un proyecto ejecutado en una "froma DevOps" puede no ser conveniente, porque en algún momento va a ir en contra de su "operaciones normales de TI", y los dos podrían no llevarse bien.

Por lo tanto, podría tener equipos que se comportan de acuerdo a los principios de DevOps, y se puede llamar un "equipo de DevOps" si sólo tiene uno. Pero es incorrecto pensar que DevOps es implementado por algún equipo dedicado a las "implementaciones de DevOps". Es importante distinguir que "el equipo que maneja DevOps para nosotros", que no es lo mismo que "un equipo que se comporta de acuerdo con DevOps". Esa es una línea super-fina, tal vez, pero es una distinción importante.

Lo que no es DevOps

Teniendo en cuenta que DevOps es una filosofía … un enfoque de gestión … y la combinación de múltiples disciplinas de TI … podría ser más fácil ver rápidamente lo que no es.

- DevOps no es ágil. Dicho esto, sus equipos podrían utilizar Agile como una metodología de desarrollo dentro de un enfoque global de estilo DevOps. Agile es ciertamente compatible con DevOps, y, al igual que DevOps, valora la mejora continua.
- DevOps no es Integración Continua. Dicho esto, CI es a menudo una parte del comportamiento de estilo DevOps. Los dos pueden estar estrechamente relacionados, de hecho - tan cerca que es difícil distinguir la diferencia. Supongo que podría argumentar que es difícil practicar la filosofía de DevOps sin usar CI, pero definitivamente puede tener CI sin comportarse como una organización de DevOps, por lo que los dos no son exactamente lo mismo.
- DevOps no es "los desarrolladores que operan". En todo caso, las operaciones están automatizadas hasta el punto en que se ejecutan en respuesta a las acciones autorizadas tomadas por otros roles, incluidos los desarrolladores.
- DevOps no es una metodología de desarrollo de software. Vea la primera viñeta, arriba. DevOps es lo que ocurre mientras

el desarrollo del software está sucediendo, y en gran parte lo que sucede cuando se desarrolla el software (o un ciclo de él). Usted todavía necesitara administrar su proceso de desarrollo de software - solo necesita usar una metodología que sea compatible con DevOps.
- DevOps no es automatización. Sin embargo, no puede tener DevOps sin automatización. La automatización es quizás el mayor beneficio que operaciones recibe de DevOps.

Además, parece ser un objetivo no declarado evitar la creación de cualquier tipo de marca registrada DevOps, o del típico libro de reglas de "cómo hacer DevOps", a la ITIL o TQM o algo así. *Este libro* ciertamente no intenta proveer "reglas". El objetivo aquí es proporcionar una cierta comprensión de lo que son los objetivos generales de DevOps..

¿Cómo se ve DevOps?

Si vamos a concentrarnos en el rol de Operaciones de TI en una organización de DevOps, resulta útil pensar en lo que realmente es un proyecto de DevOps. ¿Qué es exactamente lo que provee la operación de TI? ¿Qué capacidades necesita la organización? Vamos a tomar un aspecto de alto nivel en un proyecto al estilo DevOps, y todo lo que esto implica, y vamos a profundizar en varias partes de esto en el resto del libro.

SIN EMBARGO, quiero enfatizar que usted no puede lograr DevOps completamente dentro del equipo de Operaciones. *DevOps es pensar en todo el sistema* (una frase muy Deming), desde las personas que escriben el código hasta las personas que utilizan dicho código, y todo lo demás. El equipo de Operaciones tiene una contribución, al igual que muchos otros equipos y roles.

Hay *un montón de gente hablando de DevOps* en estos días, por lo que también hay un montón de opiniones diferentes sobre cómo debe funcionar un "proyecto DevOps". En la búsqueda de una explicación concisa y de alto nivel, me quedé bastante impresionado con una descripción de cómo Spotify[8] organiza sus esfuerzos de TI. Aunque gran parte de esa descripción se centra en cómo se organizan los desarrolladores de software, lo interesante para mí fue que el trabajo principal de sus operaciones de TI era crear unidades de automatización para que los desarrolladores pudieran implementar su código directamente en sus ambientes de QA y producción. Las operaciones, en otras palabras, facilitaron una conexión segura y administrada entre desarrolladores y usuarios de aplicaciones (servicios). Ops más o menos arregló las cosas de modo que el mismo Ops "salió del camino", dentro del marco de gestión y control de la actividad.

[8] https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/

Este es el corazón de DevOps, y si esto hace que su corazón lata más fuerte, entonces tiene que recordar que DevOps es una filosofía muy diferente de lo que ha hecho antes. En el pasado, los equipos de QA y de Operaciones eran generalmente equipos separados dentro de TI. El código pasó de los desarrolladores a QA y viceversa, hasta que QA verificó y aprobó, para que luego Operaciones lo llevará a los ambientes de producción. La intención de tener estas "puertas" entre roles era asegurarse de que nadie hiciera algo que no se suponía que debía hacer, como desplegar código no aprobado en los ambientes de producción. Esto creó varios problemas:

- Los desarrolladores se volvieron perezosos. Sabían que QA estaba revisando su trabajo, por lo que se concentraron menos en producir código de calidad. QA, a su vez, tuvo que tomar su trabajo más en serio, por lo que las organizaciones comenzaron a invertir mucho en la automatización de controles de calidad. Como resultado, la organización gastó una tonelada de tiempo y dinero permitiendo a los desarrolladores hacer sus trabajos con menor calidad. Esto no era bueno para nadie. Es evidente que las pruebas son importantes, pero el enfoque dev-versus-QA no ha sido masivamente beneficioso ni eficiente.
- La organización desarrolló una actitud "anti-nosotros", que es probablemente como su organización se comporta ahora mismo. Pero eso no es divertido. Después de todo, *se supone que todos tenemos el mismo objetivo*: ofrecer software y servicios a los usuarios, por lo que se supone que debemos estar juntos. En los peores casos, la rivalidad interdepartamental se vuelve verdaderamente tóxica, lo que lleva a que el lugar de trabajo se convierta en algo desagradable e improductivo.
- Operaciones cometió algunos errores simplemente porque *ellos no escribieron el código*, y los desarrolladores tenían poco incentivo para escribir código que fuera fácil de implementar, administrar o supervisar. Los desarrolladores simplemente lanzaron el código "al otro lado del muro" y Operaciones

tuvo que lidiar con él - aumentando la tensión entre los departamentos.

Todo esto conspiró para crear algo que es esencialmente la antítesis de DevOps. Los lanzamientos de software son más lentos, debido a la marcha implacable del código desde desarrollo hasta QA, para finalmente llegar a producción. Operaciones, básicamente, vive en el miedo de nuevo código, porque saben poco sobre él, y este no fue necesariamente diseñado para facilitar la operación. Lanzamientos más lentos significaron más presión para empaquetar más funciones en esos lanzamientos, por lo que cada lanzamiento se convirtió en un "triunfo", que simplemente empeoró el proceso.

Por el contrario, DevOps prevé la aplicación y la prestación de servicios que empujan constantemente pequeñas actualizaciones incrementales a los usuarios, con un mínimo de sobrecarga operacional. Los lanzamientos más pequeños son más fáciles de codificar y probar, y con el enfoque correcto, más seguros para empujar hacia producción de forma continua. Pero para que todo eso suceda, todos tienen que trabajar juntos. La línea entre desarrollador y operaciones tiene que ser borrosa.

En un entorno de DevOps, las cosas funcionan de manera diferente. Aquí está una mirada súper simplificada:

1. Los desarrolladores codifican y comprueban su código en un repositorio.
2. En algún momento, el código actual del repositorio se extrae y se incorpora a una aplicación.
3. Las pruebas, generalmente automatizadas y creadas por los desarrolladores, se ejecutan, incluidos los modelos individuales, las pruebas de integración e incluso las pruebas de aceptación por parte de los usuarios.
4. Si las pruebas tienen éxito, la generación se implementa automáticamente en producción (o al menos en algún ciclo de implementación).

5. Se recopilan los comentarios de los usuarios, alimentando la siguiente iteración del ciclo. Vuelva al paso 1

Algunas partes de esto pueden ser extremadamente automatizadas, y partes - como la aceptación del usuario - todavía puede ser hecha manualmente por seres humanos. El punto es disminuir las barreras entre el codificador y el usuario. Eso no significa que no haya puntos de control a lo largo del camino – para eso son las pruebas, después de todo - pero no se pone a una parte del equipo de TI como responsable de "detener" a otra parte del equipo para que no haga algo estúpido. DevOps, como una filosofía, implica que usted confía en su equipo. Si no confía en alguien de su equipo, entonces tiene un problema de recursos humanos, y debe educarlos para poder confiar en ellos, o despedirlos y reemplazarlos con alguien de confianza. Si su empresa "nunca permitiría que el código de un desarrollador llegará a producción sin que otras 30 personas lo aprobaran primero", entonces usted no puede hacer DevOps. Eso es lo que estaba escribiendo anteriormente acerca de la gestión de buy-in que es el primer paso.

La idea detrás de DevOps es, como he notado, suavizar la trayectoria entre el codificador y el usuario, de modo que las pequeñas actualizaciones incrementales de la aplicación se puedan liberar más o menos todo el tiempo. Cuando se reciben comentarios de los usuarios, los codificadores responden y luego liberan más actualizaciones.

Por cierto, aquí está una gran explicación de lo que es DevOps[9] - y lo que no es. Es un artículo largo, pero vale la pena leerlo, y notará cuánto se necesita la administración de buy-in para que todas esas cosas funcionen.

Por lo tanto, para los propósitos de este libro, necesitamos ver algunas de las cosas necesarias para hacer que el paso 4 suceda, y un poco sobre lo que se necesita en el paso 3 también. Una

[9] http://theagileadmin.com/what-is-devops/

vez más, nos centraremos principalmente en procesos y prácticas. Definitivamente *necesitará alguna tecnología para implementarlas en la vida real*, pero las tecnologías exactas que elija dependerán de su entorno específico, por lo que vamos a mantener esto un poco más abstracto por ahora.

Capacidades operacionales de un entorno DevOps

¿Cuáles son algunas de las capacidades que necesita implementar en un entorno DevOps?

Creación automatizada del entorno

En primer lugar, y posiblemente, antes que cualquier otra cosa, necesita capacidad de alistar entornos automáticamente y constantemente. Es una tarea fundamental pero no es fácil.

- Automáticamente: Significa permitir a una variedad de roles autorizados dentro de su organización configurar entornos bajo demanda, sin involucrar a ningún ser humano. Esto podría ser un entorno para desarrollo o un entorno de pruebas, y probablemente sea algo que necesitan hacer varias veces al día. También podría ser un proceso automatizado alistando un entorno en el que ejecutar pruebas de aceptación.
- De manera consistente: Los entornos que se "crean" deben reflejar con precisión el entorno de producción final. Hay dos formas de hacerlo:
  - Definir un método de creación de entornos, y utilizarlo para crear el entorno de producción, así como cualquier otro entorno cuando sea necesario. De esa manera, sabrá que todos los entornos coinciden.

- Modelar un entorno a partir del entorno de producción. A continuación, puede aplicar ese modelo a cualquier otro entorno que necesite alistar.

Tecnologías de gestión de configuración emergentes, como DSC de Microsoft, o productos como Chef, Salt, Puppet y Ansible, son ejemplos de herramientas que ayudan a implementar algunas de estas capacidades. Cuando puede escribir algún tipo de documento de configuración que describe el entorno y, a continuación, tiene una herramienta que puede implementar ese documento donde y cuando quiera, se estará acercando a la capacidad necesaria. Los containers son otra tecnología que puede ayudar en este espacio, ya que le permite abstraerse de una serie de variables, reduciendo la transformación y complejidad.

Es fácil entender por qué esta es una capacidad tan importante. Si puede garantizar que todo lo que una aplicación puede ejecutar (desarrollo, prueba o producción) es exactamente el mismo, todo el tiempo, entonces es mucho menos probable que tenga problemas para mover el código de un entorno a otro. Además, al ofrecer a otros roles, como los desarrolladores, la capacidad de generar estos entornos bajo demanda, ayuda a facilitar más pruebas en el mundo real y elimina más problemas durante la fase de desarrollo.

No quiero minimizar la dificultad de crear esta capacidad, ni tampoco ocultar los problemas que esto trae a la gestión. Los entornos requieren de recursos para funcionar, por lo tanto, las organizaciones pueden estar justificadamente preocupadas por permitir a los desarrolladores "crear" máquinas virtuales a su antojo. Pero *no estamos hablando de capacidades sin gestión alguna.* Eso es algo que me mata cada vez que entro en una discusión sobre DevOps con ciertos tipos de organizaciones. "¡Bueno, una vez que demos a los desarrolladores permisos para crear las máquinas virtuales que quieran, será el fin del mundo!" Y de esta forma presienten una derrota temprana. Pero eso no es de lo que estamos hablando.

La razón por la que DevOps tiene "Ops" al final, es porque *Ope-*

raciones no desaparece. Los desarrolladores no "toman el control". Nuestro trabajo es proporcionar a los desarrolladores un *conjunto gestionado de capacidades*. Así que sí, un desarrollador que trabaja en un proyecto debe ser capaz de "montar" una máquina virtual sin la intervención de nadie, también de ser capaz de reciclarla, es decir, eliminar y volver a crear ese entorno en el momento que se requiera. Pero eso no significa que pueda llegar a cambiar la especificación del entorno por sí mismo, ni tampoco significa que obtendrá un dominio libre de la infraestructura de virtualización. No.

Permítanme ofrecer un ejemplo realmente simplista, pero increíblemente real, de lo que estamos hablando. El servicio Elastic Beanstalk de Amazon está diseñado para crear nuevos entornos, es decir, máquinas virtuales, más o menos a la carta, en respuesta a la carga del cliente. Cada nueva máquina virtual se inicia como una copia idéntica de una imagen del sistema operativo base y cada nueva máquina virtual puede cargar contenido, como un sitio web, desde un repositorio de GitHub. Así que ahí mismo, ha creado algo de la automatización y la coherencia que necesita. Con un "botón de inicio", o en reacción a la carga del usuario, se puede automatizar la creación de nuevos entornos, y como todos provienen de fuentes conocidas estándar, este entorno será coherente..

Es muy probable que los desarrolladores necesiten cambios más allá de lo que hay en la imagen de base del sistema operativo, por lo que estos deben poder especificar elementos adicionales. Pueden establecer variables de entorno, especificar paquetes para descargar e instalar, y así sucesivamente. En el pasado, un desarrollador habría manipulado su entorno de desarrollo hasta que todo funcione, y luego con suerte comunicar los resultados de esa manipulación a alguien en de Operaciones. Ops entonces, con suerte, volvería a crear fielmente lo que el desarrollador hizo. ¿Pero se obtuvieron las versiones correctas de los paquetes? ¿Se configuraron todas las variables de entorno?

Sin embargo, en Elastic Beanstalk, los desarrolladores no sólo "modifican" el entorno. Eso es porque cada vez que una máquina virtual

se apaga, se desvanece. Cualquier retoque que se haga se pierde. En el siguiente inicio, se vuelve a la imagen del sistema operativo base. Por lo tanto, como parte del origen del proyecto en GitHub, los desarrolladores pueden especificar un archivo de configuración que enumera explícitamente todos los paquetes adicionales, la configuración del entorno o lo que sea que necesiten. Dado que esa información de configuración forma parte de la fuente GitHub, cada nueva VM creada por Elastic Beanstalk se creará con la misma configuración exacta, cada vez.

Este es un enfoque muy DevOps, y en este caso, Amazon ha asumido el papel de "Ops". Si un desarrollador quiere hacer un cambio ambiental, modifica la fuente del proyecto y luego le dice a Amazon que recicle el ambiente. Todo se apaga, y un ambiente nuevo y fresco se libera. Está completamente documentado, así que, si funciona de la manera que el dev quiere, entonces será perfecto cuando se usa para pruebas, producción o cualquier otra cosa. Y, de una manera típica centrada en la nube, Ops, es decir, Amazon, no tiene que involucrarse de ninguna manera. Han creado interfaces de automatización que permiten a cualquier usuario autorizado modificar lo que quiera.

Como una barra lateral, esta idea de DevOps es una especie de seguimiento del concepto de "nube privada". Nube privada significa simplemente ejecutar sus recursos de TI privados de una manera similar a los proveedores de una nube pública, lo que implica la automatización en el lado de Operaciones. Se basa en una forma de especificar quién puede hacer qué, y luego dejar que lo haga por su cuenta. Con un proveedor de Cloud público, los permisos consisten más o menos en "lo que se paga", pero en una situación de nube privada, los permisos pueden ser mucho más granulares o incluso completamente diferentes. Nadie sugiere que construya su propio AWS o Azure. Eso no es lo que significa nube privada. Sin embargo, verá que las capacidades de la nube privada son las mismas que debe proporcionar como persona de Operaciones, para habilitar un enfoque de DevOps dentro de su organización.

Infraestructura de desarrollo y pruebas

Como describí en el capítulo anterior, la administración de TI tradicional coloca algunas "puertas" bastante firmes entre desarrollo, pruebas y, especialmente, Operaciones. "Operaciones" es más o menos un sinónimo de "producción". En DevOps, rompemos esa relación y eliminamos las puertas. Operaciones es responsable de la infraestructura, ya sea que la infraestructura se para soporte de desarrollo, de pruebas o a los usuarios de producción. Y esas diferentes fases del ciclo de vida de la aplicación se integran mucho más estrechamente en DevOps. Algunas de las cosas de alto nivel que necesitará son:

- Repositorios de código fuente. Git es un ejemplo común en estos días, al igual que Microsoft Team Foundation Server y algunos otros. Lo importante es que las herramientas de sus desarrolladores estén estrechamente integradas con lo que haya elegido. Idealmente, estos repositorios deberían tener, o ser capaces de integrarse con, algún tipo de "codificación avanzada". Por ejemplo, el repositorio debería ser capaz de ejecutar pruebas predefinidas en el propio código antes de permitir el registro de cambios, y podría realizar una rutina automatizada de compilación y pruebas cada vez que se "detecte" código nuevo o cambios en el mismo.
- Tableros de instrumentos. Los desarrolladores y probadores necesitan tener acceso a las capacidades operativas que les han proporcionado, como la de reciclar un entorno de desarrollo virtual. Idealmente, se puede integrar esto como parte de su herramienta de gestión principal, o como un entorno de desarrollo integrado. Ser capaz de hacer clic en un botón para "compilar eso, preparar un entorno de desarrollo, cargar el código compilado y ejecutar la aplicación" es bastante potente. En los casos en que ese nivel de integración no sea

posible, entonces necesitará proporcionar alguna otra interfaz para hacer que algunas de esas actividades sean fáciles de llevar a cabo.
- Herramientas de prueba. Una cierta cantidad de pruebas tiene que ser automatizada, para que los desarrolladores pueden obtener retroalimentación inmediata, y para que las pruebas se pueden ejecutar de la manera más coherente posible.

Esa última capacidad es quizás una de las más complejas. En un enfoque ideal (aunque ciertamente no el único, e incluso este será un ejemplo simplificado), un flujo de trabajo podría ser algo como esto:

1. El desarrollador escribe código.
2. El desarrollador ejecuta código en un entorno de desarrollo "privado", realizando pruebas unitarias.
3. El desarrollador repite los pasos 1-2 hasta que esté satisfecho con el código y, a continuación, lo verifica en un repositorio.
4. El repositorio ejecuta ciertos controles de calidad, que podrían ser simplemente cosas como hacer cumplir las convenciones de codificación, antes de permitir el registro definitivo.
5. Si el check-in tiene éxito, el repositorio arranca una compilación automatizada del código. Esto se implementa en un entorno de pruebas recién creado.
6. Las herramientas de pruebas automatizadas ejecutan una serie de pruebas de aceptación en el código. Esto puede implicar proporcionar entradas específicas a la aplicación y luego buscar salidas específicas, "hackear" datos en una base de datos para probar la respuesta de la aplicación, etc. La creación de estas pruebas es realmente un esfuerzo de codificación en sí mismo, y puede ser completado por el desarrollador que trabaja en el código, o por un codificador de pruebas dedicado.
7. Los resultados de las pruebas se almacenan - a menudo como una parte del repositorio de código fuente.

8. Si las pruebas tuvieron éxito, la compilación se prepara para su implementación. La implementación puede ocurrir durante una ventana programada después de la compilación.

Puede ver que el trabajo humano aquí está casi todo del lado de los desarrolladores, que es una razón por la que la gente se refiere a DevOps como una "metodología de desarrollo de software". Sin embargo, la pieza Ops proporciona toda la infraestructura y la automatización desde el paso 4, permitiendo que una construcción exitosa se pueda mover directamente al ambiente de producción.

Obviamente, las diferentes organizaciones tendrán diferentes puntos de vista. Algunas podrían obligar a las pruebas de aceptación de usuario como un paso manual adicional, aunque Ops podría ayudar a automatizarlo. Por ejemplo, después del paso 7, puede automatizar la creación de un entorno de prueba de aceptación de usuario, implementar el código en ese entorno y, a continuación, notificar a alguien que esté listo para realizar las pruebas. Su aceptación podría desencadenar el paso a producción, o su rechazo podría regresar al desarrollador y comenzar de nuevo en el paso 1.

El punto es que *Operaciones* debe proporcionar la automatización para que esta secuencia se ejecute con la menor intervención manual como sea posible. Ciertamente, *Ops nunca debe actuar como un guardián*. No son probadores de código. Si el código pasó los puntos de control de calidad que se han definido, entonces el código está listo para ser implementado, y todo eso debe ser tan automáticamente como sea posible. Incluso el despliegue - una vez aprobado, y en cualquier horario que se haya definido - debería ocurrir automáticamente.

Se puede ver a DevOps, como una filosofía abstracta, que en realidad requiere una gran cantidad de herramientas concretas. Y tal vez usted puede observar que, debido a que las organizaciones tienen todas diferentes maneras de administrar el proceso, sería difícil para los vendedores comerciales producir esa herramienta. En realidad, no existe un enfoque de "tamaño único" para DevOps, lo que

significa que Operaciones acabará creando una gran cantidad de herramientas propias. Ahí es donde entran en juego las *tecnologías de plataforma*. Pueden proporcionar un conjunto de bloques de construcción que faciliten la creación de las herramientas de DevOps personalizadas que necesitará.

Supervisión de la experiencia del usuario final

Esta es quizás la parte más importante de una organización DevOps, y es la más fácil de pasar por alto.

Como una persona de operaciones de TI, es probable que ya esté bastante familiarizado con el monitoreo, y no se equivoca: es tan importante en DevOps como antes de DevOps. Supervisión no sólo para notificar a alguien cuando algo va mal, sino también supervisar las aplicaciones de perfil (y sus servicios de apoyo e infraestructura), para que se puedan resolver proactivamente los problemas antes de que se conviertan en graves.

Pero la definición de "seguimiento" de IT Ops a menudo no es tan inclusiva como debería ser. Tendemos a monitorear solamente las cosas que están directamente bajo nuestro control. Monitoreamos el uso de la red, la carga del procesador y el espacio en disco. Supervisamos la latencia de la red, los tiempos de respuesta del servicio y la salud del servidor. Controlamos estas cosas *porque podemos afectarlas directamente*.

Una de las mayores colaboraciones que una organización de DevOps puede tener, sin embargo, es monitorear *la experiencia del usuario final*. Es algo que nosotros, como gente de TI, no podemos tocar directamente, pero si la razón de ser de TI es entregar aplicaciones y servicios a los usuarios, entonces la experiencia del usuario final de esas aplicaciones y servicios es literalmente la única métrica que importa. ¿Por qué medimos la latencia de la red? Porque

contribuye a la experiencia del usuario. ¿Por qué medimos el tiempo de respuesta del servicio? Experiencia de usuario. Intentamos medir *indirectamente* la experiencia del usuario final, porque a menudo no tenemos forma de medirla *directamente*.

La filosofía de desarrolladores y operaciones de DevOps llega a la cumbre con la supervisión de la experiencia del usuario final. Los desarrolladores deben crear aplicaciones con la capacidad de rastrear la experiencia del usuario final. Por ejemplo, cuando una operación común está a punto de comenzar, la aplicación debe rastrear la hora de inicio y, a continuación, seguir la hora de finalización. Cualquier paso importante entre ambos debe recibir una marca de tiempo, también, y esa información debe registrarse en algún lugar. En Operaciones, necesitamos proporcionar un lugar para que el registro - ese artefacto de rendimiento - viva, y necesitamos proporcionar una forma para que los desarrolladores accedan a él. Tenemos que determinar a que se parece el rendimiento "normal", y definir una ruta para realizar el seguimiento de las *afectaciones en esa línea base*. Operaciones pueden ser responsables de la supervisión en sí, pero los desarrolladores, en su código, deben proporcionar la instrumentación para controlar lo que más importa.

Si los números de la experiencia del usuario final comienzan a disminuir -digamos, el tiempo que se tarda en realizar una consulta y mostrar los resultados comienza a hacerse más y más largo -, podemos buscar una instrumentación más detallada y ver si podemos encontrar la causa. ¿Es latencia de la red? ¿Tiempo de respuesta del servidor? ¿Alguna otra correlación que pudiera apuntar a una causa? Pero al medir directamente *lo que nuestros usuarios experimentan*, contamos con una métrica de alto nivel infalible que representa lo más real que podemos tener en el radar.

Estoy extendiéndome en el tema de monitorear de la experiencia del usuario final no sólo porque es útil e importante, sino también porque es uno de los ejemplos más fáciles de comprender acerca de lo que se trata DevOps. Tradicionalmente, los desarrolladores *se han preocupado* por la experiencia de los usuarios (en teoría),

pero están extremadamente *desconectados* de ella. Operaciones en cambio, está muy conectado a lo que los usuarios experimentan (después de todo ellos reciben las llamadas al Help Desk), pero son relativamente impotentes para aplicar medidas correctivas. A través de la colaboración que impulsa la filosofía de DevOps, sin embargo, los desarrolladores y el personal de operaciones pueden unirse para hacer un trabajo colectivo mucho mejor.

Habilidades IT Ops en un entorno de DevOps

Digamos que ha decidido, al menos en teoría, ayudar a llevar a su organización a una posición de DevOps. Ha leído acerca de algunas de las capacidades de alto nivel que usted, como Operador, debe proporcionar a la organización.

¿Cómo lo hace?

En una palabra, "pegamento".

Lo diré de nuevo: *DevOps es una filosofía.* La contabilidad sigue siendo un buen ejemplo. La industria de la contabilidad está de acuerdo, más o menos, en lo que constituye una buena contabilidad, y de ahí es de donde provienen los PCGA. Del mismo modo, la industria de DevOps está definiendo lentamente de que se trata un "DevOps bueno"

Pero cada organización lo hace a su manera. Observe cómo cada organización maneja su contabilidad, en detalle, y encontrará un montón de diferencias con otras organizaciones. Tal vez los auditores trabajen de formas diferentes, o tal vez un rol de trabajo diferente es responsable de diferentes funciones de contabilidad. Algunas empresas necesitan una contabilidad bastante simple, mientras que otros necesitan una contabilidad increíblemente compleja que exige cientos de personas que trabajan las veinticuatro horas del día. Aunque todos ellos operan con *los mismos principios*, sus implementaciones varían ampliamente.

Así es con DevOps.

En una organización pequeña, la contabilidad puede ser lo suficientemente simple como para que las herramientas disponibles en el mercado, como Quickbooks, sean suficientes. En ese tamaño de

una organización, DevOps ni siquiera podría existir, porque una empresa de ese tamaño simplemente podría no hacer ningún "dev" para empezar. En una empresa masiva y multidepartamental, la contabilidad podría incluir herramientas "disponibles" que requieren meses y meses de personalización y ajustes. Del mismo modo, DevOps en esa misma organización podría implicar herramientas personalizadas que utilizan bloques genéricos de construcción … y un montón de pegamento personalizado.

Proporcionar la infraestructura operacional para una organización de DevOps puede ser hacking en su mejor momento. Sí, usted encontrará un montón de productos y tecnologías disponibles en el mercado … pero muchos de ellos sólo le llevarán hasta cierto punto en las metas de su organización. Después de eso, tendrá que hacer mucho de personalización, y poco de "pegado" de herramientas diferentes clases, además de algo de "hacking" alrededor para juntar las piezas. Probablemente siempre será así, así como todavía es el caso de nuevos despliegues de herramientas de contabilidad que por lo general toman meses y meses. Nada fuera de una plataforma podrá cubrir todas las necesidades de cada organización, por lo que simplemente tendrá que estar preparado para hacer algo de personalización, algo de hacking y un poco de pegado.

Con eso en mente, ¿cuáles son las habilidades adecuadas que se deben tener?

- Capacidad de aprender rápidamente. Tendrá que dominar nuevos productos y tecnologías sobre la marcha.
- Creatividad. Tendrá que pensar en soluciones inteligentes para evitar obstáculos. No espere que todo "funcione" - no lo hará.
- Conocimiento profundo de su plataforma(s). Ya sea que esté trabajando en Microsoft Windows, una distribución de Linux o alguna otra plataforma, necesita conocer profundamente cómo funciona, porque va a interactuar con ella a muy bajo nivel.

- Scripting. Va a necesitar ser fluido en lenguajes de programación de sistemas ("scripting") usados en su plataforma, porque ese es el "pegamento" que usará para adherir diferentes tecnologías en una solución coherente y personalizada.

Este material de DevOps no es para principiantes, ni para débiles de corazón. Es por esto que, en mi propia creencia personal, las empresas crean títulos de trabajo como "DevOps Engineer". La mayoría de la comunidad de DevOps con bastante razón se asusta por títulos de trabajo como ese, porque a menudo son una demostración de que *alguien en la organización no lo entiende*. DevOps no es un rol de trabajo. Sin embargo, en una organización que practica DevOps, hay ciertamente algunas habilidades que será muy práctico contar con ellas, especialmente en el lado de Operaciones. Alguien que posea esas habilidades podría ser llamado "Ingeniero de DevOps", que es quizás menos engorroso que "Una persona de TI que sabe lo suficiente para pegar todos esos bits de tal manera que podamos obtener las capacidades de DevOps que necesitamos". Eso sería una gran tarjeta de presentación. "DevOps Engineer" es probablemente también un título menos vergonzoso para los compañeros de trabajo que "El Chico listo de IT que necesitamos", que al final suele ser el caso.

La gente de IT Ops que trabaja para proporcionar las capacidades compatibles con DevOps es a menudo la gente más experimentada y más inteligente del equipo. Tienen más experiencia y conocimiento, y son a menudo los más ansiosos de hacer frente a este desafío.

Por cierto, fíjese cómo lo expresé. "… trabajar para proporcionar las capacidades compatibles con DevOps …" fue una frase deliberada. Una organización que practica DevOps necesita capacidades específicas, y la parte de Operaciones proporciona algunas de ellas, en estrecha colaboración con el lado Dev. Pero esto no significa que usted vaya a tener un "Departamento de DevOps", eso no tiene sentido. "DevOps Engineer" como un título de trabajo es sólo legítimo si significa "Ingeniero que ayuda a proporcionar nuestras

capacidades relacionadas con DevOps". DevOps *no es algo que simplemente se haga*. Es algo *en lo que usted cree* y que a su vez le impulsa a hacer las cosas. Si usted cree en DevOps, su organización necesitara comportarse de cierta manera, y necesitara ciertas herramientas para apoyar esos comportamientos.

También hay un conjunto de nuevas habilidades de desarrollo que necesitan ser introducidas en un entorno DevOps. Los desarrolladores tienen que centrarse más en que el código se pueda ser desplegado, supervisado y administrado de una manera centrada en DevOps. Por ejemplo, en la mayoría de los entornos basados en Windows, los desarrolladores suelen utilizar herramientas incluidas en Visual Studio para crear paquetes de Windows Installer. Esos paquetes no siempre han sido fáciles de desplegar de forma automatizada, porque pueden haber requerido (o se creían necesarios) privilegios de administrador u otros elementos que simplemente hicieron que el despliegue del código fuera difícil e incluso peligroso. Para "hacer" DevOps eso tiene que cambiar. Operaciones debe proporcionar a desarrollo la capacidad de desplegar el código de forma transparente en ambientes de producción, pero desarrollo necesita escribir código que admita ese modelo. La carga recae en ambos grupos, como un equipo combinado, no sólo en Operaciones.

Planificar para el fracaso

> "Espera un minuto," puedo oírte diciendo, "!desplegar el nuevo código directo en producción es lo que causa todos los problemas!"

De acuerdo. *Cualquier tipo de cambio tiene el potencial de crear problemas*. El punto de DevOps - y más particularmente el papel de Operaciones en DevOps - *es crear un entorno donde se puede fallar rápidamente, y arreglar con la misma rapidez* (gracias a Chris Hunt por eso). Si DevOps significa desplegar constantemente pequeños

pedacitos de código, entonces debe estar preparado para - en el lenguaje de Facebook - "moverse rápido y romper las cosas". Con el tiempo algún despliegue va a ser problemático, por lo tanto el papel de Operaciones no es ralentizar las cosas para evitar el problema, sino más bien golpear el problema duro y rápido. La virtualización, como un ejemplo, nos da la capacidad de "revertir" rápidamente entornos operativos enteros a un "buen estado conocido", haciendo que la perspectiva del fracaso sea un poco menos aterradora. *Planificar para el fracaso*, en lugar de tratar de evitar el fracaso por completo.

Pregúntese si usted es el tipo de persona que rutinariamente planea fracasar. Por ejemplo, en cada vuelo de avión que tomo, tengo un juego de ropa de repuesto en mi equipaje de mano, aunque sea sólo mi bolsa para la computadora. Tengo un pequeño desodorante, porque es un artículo no incluido en los paquetes de amenidades de las aerolíneas. Supongo que habrá un fracaso en el viaje, y tengo planes sencillos para mitigar ese fracaso. Sin embargo, pocas personas toman estos sencillos pasos, y cuando el fracaso ocurre, se enojan, estresan, se sienten incómodos, incluso cuando las causas del fracaso están completamente fuera su control, como el clima por ejemplo. Planeo esperas más largas que la mayoría de la gente - normalmente 2 horas en cada aeropuerto de cada país - y con frecuencia soy capaz de evitar un fallo en mi viaje debido a ese margen adicional.

En un entorno DevOps, tiene que aceptar que se producirá un error. Su esfuerzo debe ir menos en la prevención de ese fracaso - especialmente poniendo "puertas" que consumen tiempo como una pared entre los codificadores y los usuarios - y en su lugar poner el esfuerzo en poder iterar y recuperarse rápidamente en caso de ser necesario. En un verdadero equipo de DevOps, cuando en una liberación se encuentra un bug no significa volver a la última copia estable conocida. El verdadero significado debe ser poder hacer una nueva liberación con las correcciones mucho más rápido. Eso es avanzar en lugar de retroceder, y tener la capacidad de hacerlo es

el sello principal de una organización preparada para DevOps.

Operaciones como desarrollo

Hay un cierto temor al respecto de tener un equipo DevOps como apoyo, y es que el equipo de Operaciones se convierta en una especie de equipo de desarrollo de propósito especial. Este temor, de hecho, crea uno de los mayores malentendidos acerca de DevOps: creemos que DevOps significa "Operaciones convirtiéndose en codificadores".

DevOps no significa que Operaciones se convierta en un grupo de desarrollo. Significa que Operaciones trabajara para suavizar el camino entre el codificador y el usuario. Resulta que la forma más sencilla de operar es automatizando, y la forma más común de automatizar suele implicar cierta codificación. Por lo tanto, DevOps normalmente ocasionará que Operaciones tenga que codificar, al menos hasta cierto punto.

La mayoría de los sistemas operativos que Operaciones administra tiene algún lenguaje de programación diseñado para facilitar la automatización operacional. En Linux, por ejemplo, Perl y Python son lenguajes de scripting muy comunes. En Microsoft Windows, PowerShell ha asumido ese papel. Así que no estamos hablando de programación en C, C++, C#, u otro lenguaje de programación "profunda". Se trata más bien de "scripting", que por lo general en un lenguaje de nivel superior que está diseñado para tareas de automatización operacional. Como mencione en el capítulo anterior, la habilidad principal que Operaciones debería proporcionar a un entorno DevOps es la capacidad de automatizar en un lenguaje de scripting apropiado para su ambiente.

Pero una vez que Operaciones comienza a producir unidades de automatización, es decir, código, las Operaciones mismas necesitaran

comenzar a actuar como una tienda de DevOps. Esas unidades de automatización son la aplicación que el codificador (Ops) entrega al usuario (en este caso, otros roles en el equipo de TI). Así que Operaciones necesita las herramientas y los enfoques de gestión que permitan liberar rápidamente su código, probarlo y desplegarlo en producción. Como los usuarios (por ejemplo los desarrolladores) generan constantemente nuevas necesidades Operaciones deben estar preparado para responder a esas necesidades.

Todo este concepto es a menudo uno de los mayores obstáculos para fomentar una mentalidad de DevOps en cualquier organización, especialmente en aquellas que están fuertemente apoyadas en ambientes de Microsoft Windows. El obstáculo se debe a que los administradores de Windows, en general, no han tenido la necesidad de "aprender a automatizar", en gran parte porque el sistema operativo sólo comenzó a ofrecer esta capacidad en 2006 y solo ofreció una capacidad significativamente mejorada hasta 2012. Los administradores entonces como no han tenido las herramientas, no han aprendido las técnicas. El cambio siempre es aterrador para algunas personas (y para algunas organizaciones), por lo tanto, moverse de una administración basada en GUI (que no admite DevOps) a una administración basada en código (que sí se admite) puede ser aterrador.

Muchos administradores, una vez más, en un ambiente Windows, están acostumbrados a utilizar herramientas GUI para la administración de sus entornos. Tal vez se quejen de que las herramientas no funcionen de la manera que quieren, pero generalmente están lo suficientemente cerca y es por ello que las utilizan. Pasar a un mundo centrado en DevOps, sin embargo, introduce nuevas variables. ¿Qué tipo de código se está entregando? ¿Qué metodología utilizan los desarrolladores? ¿Cuáles son los problemas de producción en torno a la estabilidad y la disponibilidad? ¿Cuánto espacio hay para el error? ¿Qué tipo de ventanas de mantenimiento están disponibles? ¿Cómo se comunica con la base de usuarios? El gran número de variables significa que casi todas las organizaciones

son únicas, lo que significa que las herramientas disponibles en el mercado no van a ser suficientes para resolver su problemática. Consecuentemente, DevOps casi que exige que Operaciones construya sus propias herramientas y procesos, generalmente "pegando" algunas tecnologías de plataforma. Eso es lo que he tratado en el capítulo anterior, aunque desde una perspectiva ligeramente diferente.

Ese proceso de "pegamento" es donde Operaciones se desvía hacia su propio desarrollo. Es posible que utilice el Administrador de máquinas virtuales de Microsoft System Center para administrar su infraestructura virtual, pero escribirá algún código para que haga lo que desee de acuerdo con sus procesos particulares. Puede utilizar Chef para manejar la configuración declarativa de entornos de máquinas virtuales, pero escribirá algún código para decirle a Chef exactamente qué es lo que desea y para administrar los elementos personalizados que sólo existen en su entorno.

Otro resultado de este enfoque de DevOps es que, una vez que se pone en ello, comienza a tratar su infraestructura como código, y comienza a abordar la gestión de la infraestructura de una manera más ágil. La virtualización, en particular, ha hecho esto tremendamente fácil, porque podemos eliminar y volver a crear entornos masivos con el solo toque de un botón. ¿No le gusta la configuración actual del entorno? No hay problema: modifique el documento de configuración declarativa y recicle el entorno. ¿No está contento con el resultado? Repita. Reconfigurar el entorno puede (y debe) ser tan fácil como modificar una estructura similar a un código, así como modificar una aplicación es tan fácil como cambiar el código. En otras palabras, una vez que utilice código, o algo parecido, para describir cómo debería verse su entorno, básicamente está tratando la infraestructura como código. Las metodologías de desarrollo como Agile y Lean comienzan a convertirse en una opción para gestionar infraestructura… y de repente, usted ya se está convirtiendo en DevOps.

Combinar mentalmente con todos estos conceptos - infraestructura

como código, Operaciones como codificadores de pegamento - realmente abre algunas posibilidades. Ya no está limitado a este enfoque de "grandes proveedores", donde tiene que encontrar una pila de proveedores que satisfaga todas sus necesidades (lo cual nunca fue realmente práctico). En su lugar, se siente cómodo utilizando los componentes de varios proveedores según sea necesario, porque crece confiado en su capacidad de pegarlos todos juntos en la estructura que necesita.

DevOps no excluye a nadie

Porque … bueno, porque se llama Dev Ops, aunque a menudo existe esta sensación de que la filosofía excluye la seguridad … o la infraestructura de red … o los diseñadores gráficos … o alguien.

No lo hace.

Si DevOps significa cualquier cosa, significa que todos colaboran. Lo que no significa es que alguien sea vetado de participar. Por ejemplo, IT Security no puede entrar y decir, "no hay manera de que podamos automatizar el despliegue de esa aplicación debido a la seguridad". Eso no es colaboración, es obstruccionismo. Lo que pueden decir es "para automatizar el despliegue de esa aplicación en particular, necesitamos asegurarnos de que xyz están sucediendo". Seguridad, Desarrollo y Operaciones pueden trabajar juntos para automatizar esos requisitos, ayudando a asegurar que todas las veces se lleven a cabo de forma consistente. Seguridad gana porque sus preocupaciones se cumplen como parte del proceso. Desarrollo gana porque obtiene una mejor comprensión de las preocupaciones de seguridad. Operaciones gana porque deja de ser el intermediario que tiene que reconciliar los problemas de todos.

Es por esto qué DevOps no puede funcionar sin la gestión de buy-in desde un nivel muy alto. Al igual que el CEO y CIO (o CTO). Las partes de su empresa que tradicionalmente han trabajado en sus propios mandatos necesitan renunciar a sus antiguos estilos y trabajar juntos. "No" nunca es la respuesta. Es "así es cómo". Eso, estoy seguro de que usted ya lo imagina, puede ser enormemente difícil, o políticamente incorrecto, en algunas organizaciones. Y ahí es donde la gente falla en DevOps.

Una lista de lectura DevOps

Muchas gracias a Chris Hunt (@cdhunt en Twitter) por proporcionar esta lista de lecturas sugeridas.

The Phoenix Project by Gene Kim
The Goal by Eliyahu M. Goldratt
It's Not Luck by Eliyahu M. Goldratt
The Checklist Manifesto by Atul Gawande
Thinking in Systems: A Primer by Donella H. Meadows
Lean Enterprise: How High Performance Organizations Innovate at Scale by Jez Humble
Becoming a Technical Leader: An Organic Problem-Solving Approach by Gerald M. Weinberg
Theories of Work: How We Design and Manage Work[10] by David Joyce
Quiet: The Power of Introverts in a World That Can't Stop Talking by Susan Cain
Continuous Delivery by Jez Humble and David Farley
Test Driven Development by Kent Beck

[10] http://www.theoriesofwork.com/