

*Una guía de MCPRO*

# DEV OPS

*Tecnología,  
herramientas y  
plan de carrera*

“

*DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.*

Bass, Weber y Zhu

# Sumario

**¿Qué es DevOps?** Pag. 04

**¿Cómo me convierto en un ingeniero DevOps?** Pag. 09

**Tecnologías y conceptos que debes conocer** Pag. 11

Microservicios

Contenedores

Automatizadores

Git

Orquestación

Desarrollo y entrega continuos (CI/CD)

Producto Mínimo viable

**Preguntas frecuentes** Pag. 19

¿Es lo mismo que Lean y Agile? ¿Son compatibles?

¿Qué programa / solución necesito para desarrollar con DevOps?

¿Tiene aplicación DevOps en IoT? ¿Y en cloud?

¿Puedo transicionar un proyecto estándar a DevOps?

**Más allá de DevOps** Pag. 23

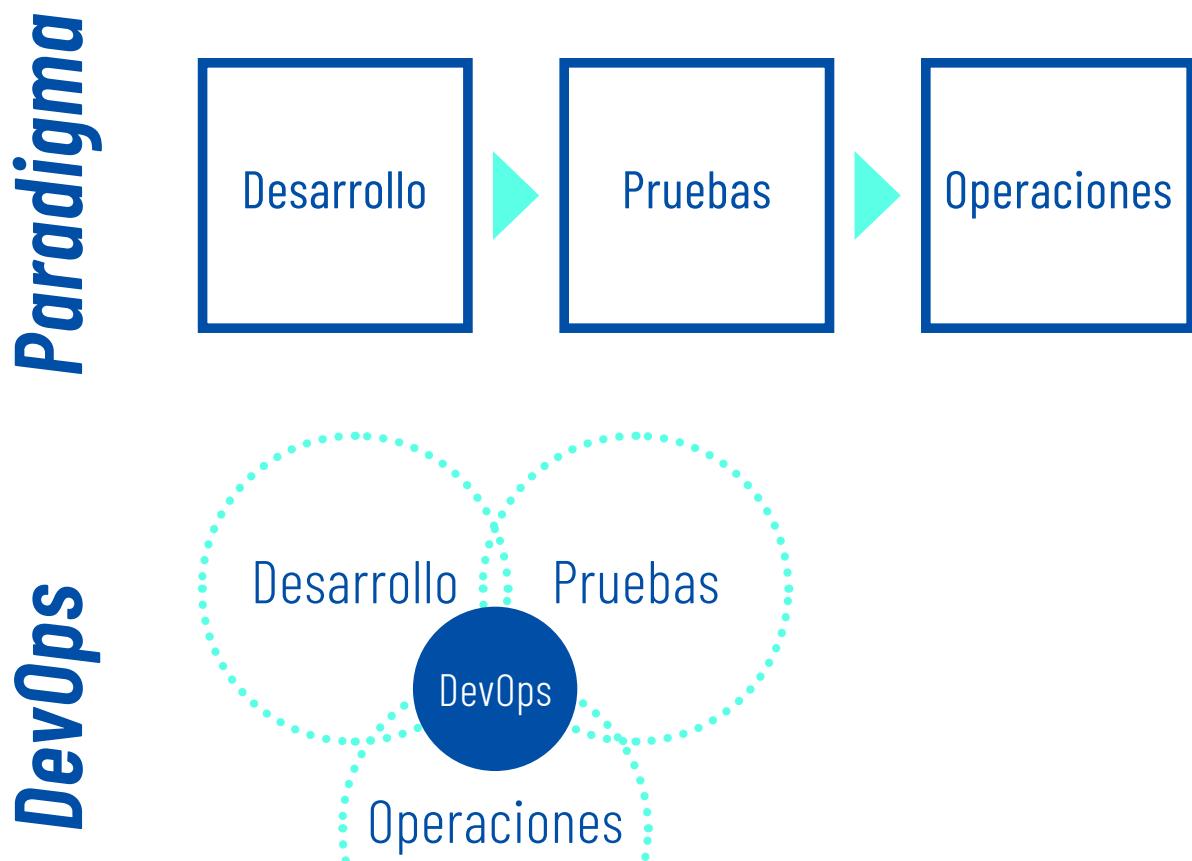


# ¿Qué es DevOps?

Hay muchas definiciones (basta con hacer una búsqueda en Internet para comprobarlo) de DevOps, y gran parte de ellas se ajustan bastante al concepto: una filosofía, una cultura, una metodología de trabajo... podemos elegir la definición que más nos guste.

Si buscamos una definición más académica. DevOps es un conjunto de prácticas, en el desarrollo de software, destinadas a mejorar los procesos, con el fin de reducir el tiempo de publicación y, al tiempo, garantizar la fiabilidad de los desarrollos.

Para tal fin, se rompe con el paradigma tradicional, en el que los equipos de trabajo realizan sus tareas de forma aislada, y en su lugar se establece una cultura de colaboración constante. Esto no solo permite acelerar la velocidad de los procesos, además se traduce en una mayor integridad estructural de los desarrollos, ya que la colaboración entre equipos ofrece un resultado final mucho más cohesionado.



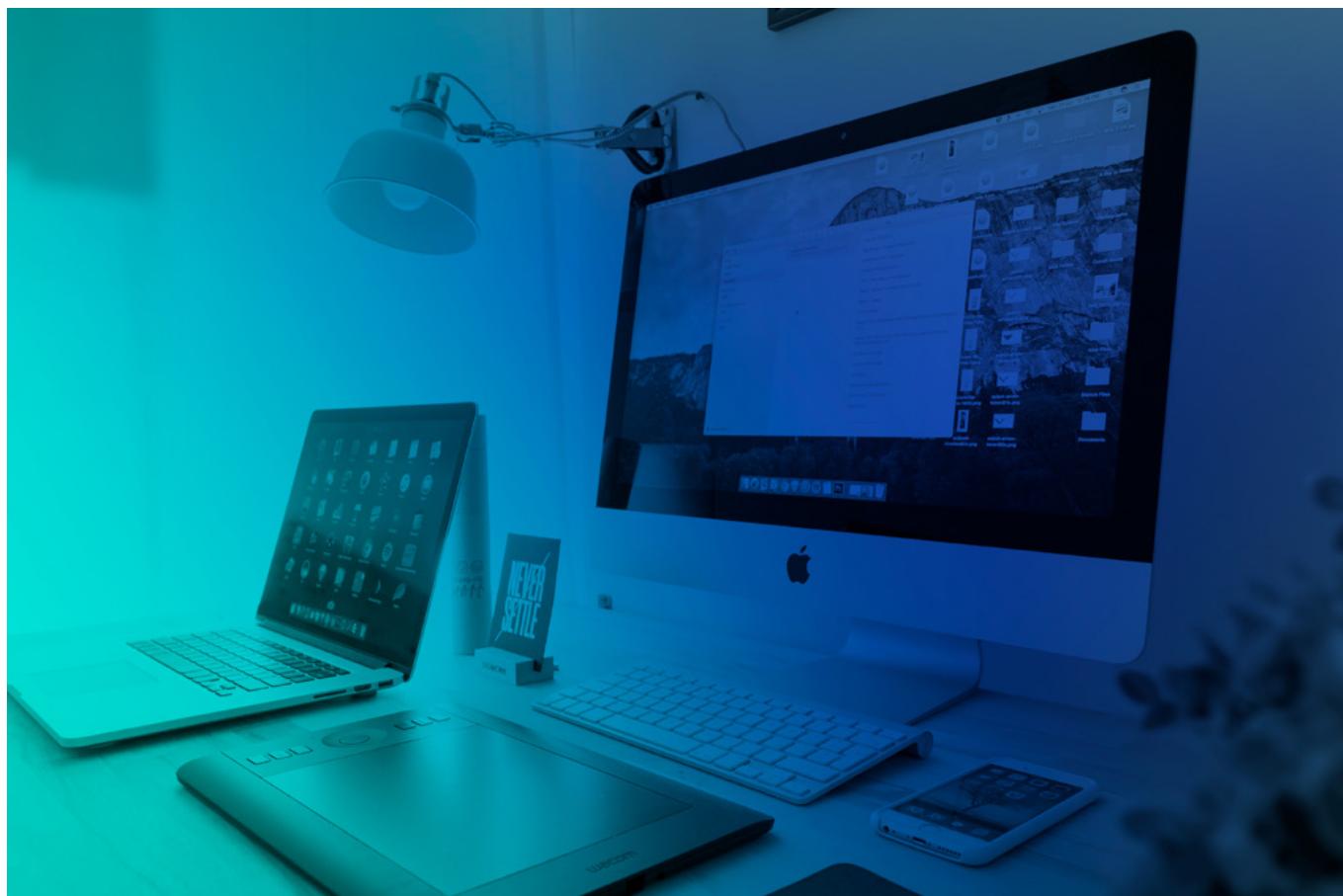
Para llevar a la práctica este modelo de trabajo, DevOps divide las tareas del mismo en varias categorías. Es importante aclarar algo: existen diversas corrientes de pensamiento, por lo que es probable que, si buscas documentación al respecto, puedas encontrar diferentes modelos. No obstante, la base subyacente es similar en todos ellos, así que si vas a acometer un proyecto y quieras aplicar la filosofía DevOps al mismo, podrás elegir el modelo que más se adecúe a tus preferencias ya que, en todos los casos, y con mayor nivel de detalle, se mantiene el modelo desarrollo, pruebas, producción.

Aunque, así visto, puede parecer que hablamos de un ciclo de desarrollo bastante semejante al tradicional, la clave se encuentra en la interacción entre las distintas categorías, la clave es que no se llevan a cabo de manera escalonada en lo referido al conjunto del proyecto. En su lugar, simultáneamente se

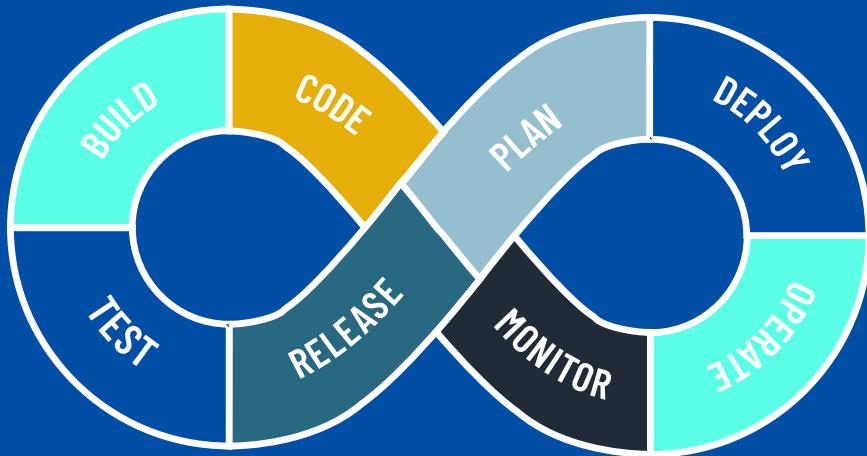
pueden estar desarrollando las siete fases en distintos elementos que integran el desarrollo.

Más adelante hablaremos de conceptos como contenedor, microservicio y demás elementos claves de DevOps, y también de integración y distribución continuas (CI/CD), pero por el momento lo importante es entender que un gran proyecto se puede subdividir en entregables más pequeños, bien integrados entre sí, y que permiten una entrega continua de mejoras, actualizaciones de seguridad, etcétera.

Por otra parte, y también dentro de la filosofía de DevOps, no existe un entorno específico para el desarrollo de proyectos. En su lugar, se recurre a varios conjuntos de herramientas, denominados **toolchains**, con los que abordar cada una de las tareas. En algunos casos son soluciones para una tarea concreta, mientras que en otros casos pueden ser empleadas en varias.



# *Distintos modelos de DevOps*



Aunque la filosofía de DevOps es común y los objetivos siempre son los mismos, no existe un único modelo de organización de las tareas. En su lugar, hay varios modelos de uso bastante extendido, además de otros más específicos, dedicados a áreas concretas de IT en las que el despliegue tiene algunas particularidades.

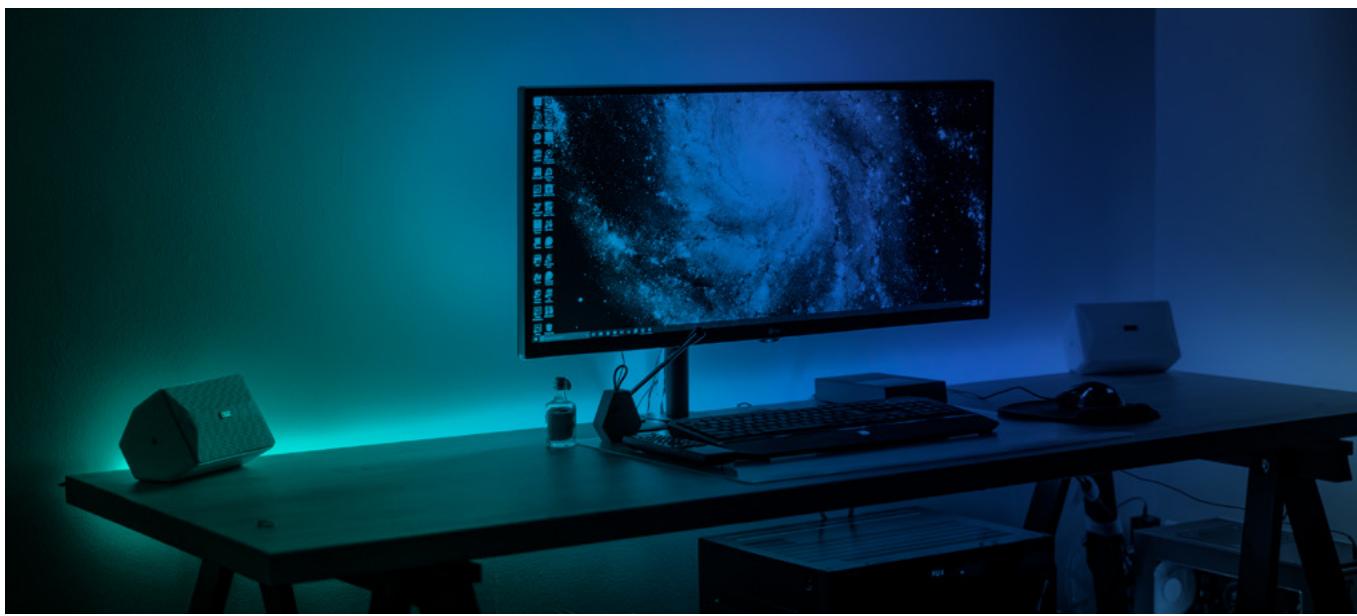
El modelo más común, y que seguro que ya has visto en alguna ocasión, es el que se representa con la imagen de un símbolo de infinito dividido en ocho partes. En el mismo, el ciclo se compone de: Plan (Planificación), Code (Código), Build (Construcción), Test, Release (Lanzamiento), Deploy (Despliegue), Operate (Emplear), Monitor (Monitorización) y de nuevo, conexión con Plan, ya que el desarrollo es, y así se entiende, continuo.

Otra división en siete pasos, bastante común (es más, es la que se puede encontrar,

por ejemplo, en Wikipedia), es Coding (Codificación), Building (Construcción), Testing (Prueba), Packaging (Empaquetado), Releasing (Lanzamiento), Configuring (Configuración) y Monitoring (Monitorización).

Y, si dedicas unos minutos a buscar en la red, encontrarás otros muchos modelos con más o menos fases. En cualquier caso, como puedes comprobar, la organización es similar en todos ellos.

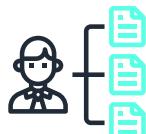
Lo más importante a la hora de poner en marcha un proyecto en cuya gestión quieras aplicar la filosofía DevOps, es que busques el modelo que más se ajuste a tus necesidades, teniendo además en cuenta que, si es necesario, puedes integrar otras fases en el ciclo si lo necesitas, como aprovisionamiento, análisis de impacto, auditoría, etcétera.



# ¿Qué es DevOps?

Esta es, lógicamente, la pregunta que se hacen muchos profesionales al plantearse aplicar esta filosofía en un nuevo proyecto. Porque, claro, dar el salto desde un modelo conocido y que resulta operativo, a uno nuevo del que

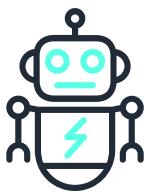
todavía no sabemos demasiado puede dar un poco de vértigo. Sin embargo, si analizamos en profundidad las ventajas que nos puede reportar esta transición, seguramente lo tendremos bastante más claro a la hora de dar el salto.



**Optimización del flujo de trabajo.** Bajo el paradigma tradicional, con múltiples equipos poco o nada conectados entre sí, es relativamente común que el flujo de trabajo no sea especialmente ágil. El establecimiento de múltiples conexiones entre los mismos, sumado a una mayor división de los entregables (en los proyectos tradicionales se tiende a agruparlos para disminuir el número de despliegues), permite que los profesionales de los diversos equipos siempre estén realizando avances en su área de responsabilidad del proyecto.



**Mejora en la publicación de versiones / mejoras.** En DevOps, al igual que en otras filosofías ágiles, se evita la tendencia de agrupar entregables para realizar grandes actualizaciones. En su lugar, se opta por ir realizando cambios y mejoras de manera más constante. De esta manera los usuarios no tienen que esperar tanto tiempo desde que se planifica un cambio o mejora, hasta que pueden disfrutar de la misma.



**Respuesta ante el feedback.** De manera adicional a lo mencionado en el apartado anterior, el equipo puede obtener feedback en una fase más temprana del desarrollo de nuevas funciones. De esta manera, si en un primer despliegue de un cambio o novedad se detecta algún problema con el mismo en el entorno de producción, se podrán efectuar los cambios necesarios antes de proseguir con su desarrollo. Esto, en no pocas ocasiones, se traducirá en un gran ahorro de tiempo de trabajo.



**Tiempo de respuesta ante incidentes.** Agilizar el modelo de desarrollo y la filosofía de trabajo de los equipos redundante, sin duda, en una mayor rapidez a la hora de desplegar una actualización o realizar un rollback (volver a una versión anterior) ante un problema de funcionamiento o de seguridad. Los proyectos DevOps suelen apoyarse en repositorios Git (hablaremos de ellos más adelante), por lo que dar “un paso atrás” es un proceso, por lo general, rápido y sencillo.



**Evolución continua.** Quizá la mejor manera de definir DevOps sea con la palabra “continua”. Y es que, al analizar el flujo de trabajo habitual de una implementación de este paradigma, veremos que las diversas tareas del mismo se pueden agrupar en cuatro grandes bloques: integración continua, testeo continuo, entrega continua y monitorización continua. Esto, sin duda, supone un esfuerzo extra en la gestión, pero a cambio ofrece unos niveles de dinamismo inimaginables en desarrollos no ágiles.



# ¿Cómo me convierto en un ingeniero DevOps?

Como ya hemos visto anteriormente, DevOps no es una tecnología, ni un conjunto de ellas (aunque obviamente se emplean diversas tecnologías en su aplicación). Además, la implantación y el funcionamiento del modelo dependerá de los responsables de los proyectos, no de los desarrolladores del mismo. No obstante, eso no significa que estos puedan permanecer ajenos al modelo de trabajo elegido, puesto que quizás tendrán que cambiar algunos aspectos de su metodología de trabajo.

Así pues, cuando hablamos de un profesional de DevOps, lo más correcto es hablar de ingenieros con capacidades específicas en relación con el paradigma. Cualquier persona implicada en la gestión de proyectos, tendrá responsabilidades en las que el modelo será de aplicación.

Con esto claro, sí que es necesario que, además de conocer y entender el modelo para poder aplicarlo, el profesional domine ciertas habilidades, que revisamos a continuación:



**Programación.** Un responsable de proyecto no tiene, por norma general, que escribir código. Sin embargo, sí que debe tener una base sólida en lenguajes de programación como Java, Python, Perl o Ruby. También le ayudará conocer PHP, pero mucho menos. Esto será clave tanto para la planificación inicial como para el desarrollo de guiones y la gestión de las integraciones de los diversos elementos.



**Aprovisionamiento y despliegue.** Es cierto que cuando hablamos de proyectos de software, tendemos a dejar un poco de lado lo relacionado con el hardware. Sin embargo, en muchos proyectos el ingeniero se enfrentará a soluciones híbridas que dependen de múltiples sistemas de almacenamiento, servidores (internos y externos), microservicios y contenedores en ejecución en distintos entornos... Un sólido conocimiento en estas áreas es imprescindible.



**Métricas.** Una de las fases fundamentales en cualquier proyecto en el que se aplique DevOps es el control constante del rendimiento de cada elemento del mismo, de cada versión desplegada, de los componentes cuando todavía están en fase de desarrollo. Es clave recopilar y gestionar esa información para realizar ajustes en recursos, correcciones en las versiones en staging y poner esta información a disposición del equipo para optimizar las iteraciones.



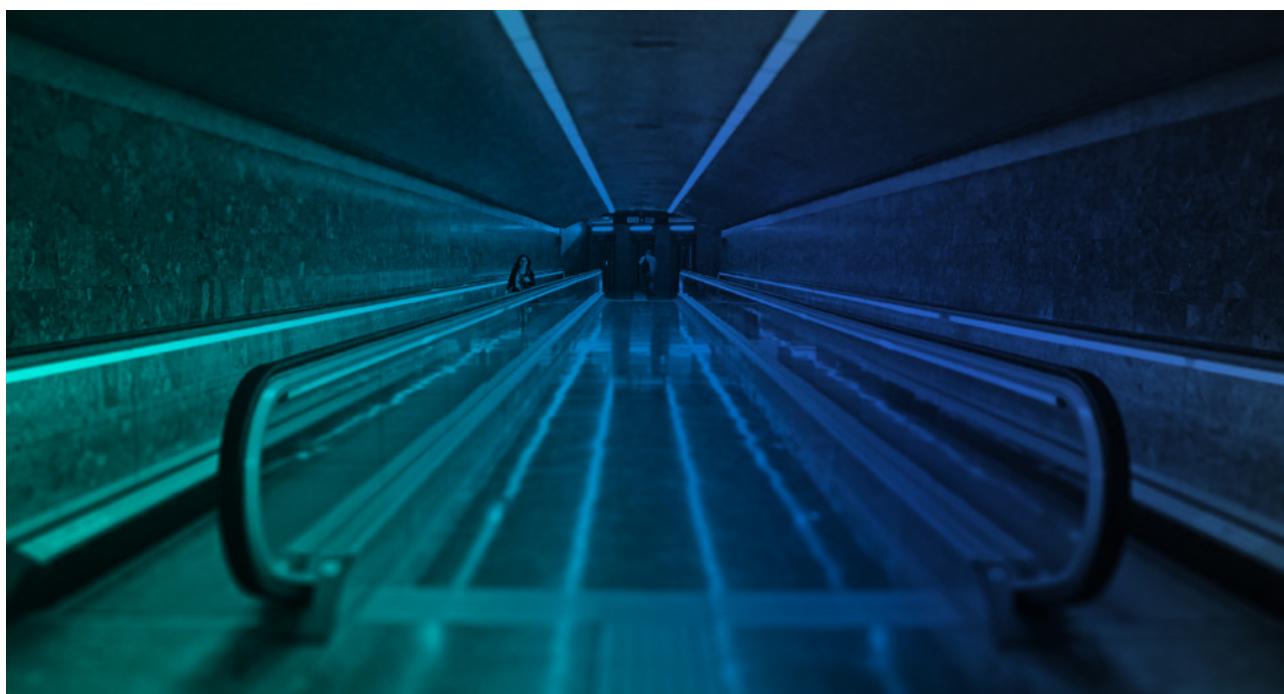
**Organización y comunicación.** Las capacidades de organización son claves en cualquier proyecto, pero esta necesidad se incrementa en un escenario tan dinámico como el que plantea DevOps. La integración continua hace necesaria una coordinación entre equipos que no existe en el modelo clásico. Y, de manera complementaria (pero igual de necesaria), una comunicación eficaz tanto con otros responsables del proyecto y la organización, como con los miembros del equipo, es clave para que todos los profesionales implicados remen siempre en la dirección correcta.



**Solución de problemas.** Tarde o temprano, en cualquier proyecto, surge algún problema. Y, como ya decíamos antes, en un modelo tan dinámico como éste, eso se puede traducir en retrasos, errores e incluso conflictos. El responsable de proyectos debe tener una buena base en la gestión de estas incidencias, para lograr minimizar su impacto y evitar (o, al menos, reducir al mínimo) sus potenciales consecuencias.



**Formación constante.** Sí, esto es algo que se aplica, en general, a los profesionales de IT. Sin embargo, tiene un especial sentido cuando hablamos de dar el salto del paradigma clásico a DevOps. No hay que olvidar que, a diferencia de otros roles en los que es necesario el conocimiento de unas pocas herramientas, cuando hablamos de DevOps tendremos que conocer y saber trabajar con, al menos, siete tipos de toolchains, compuestos a su vez de múltiples herramientas que se apoyan en distintas tecnologías. El ingeniero DevOps debe ser, en este aspecto, "enciclopédico", y mantenerse al día de las nuevas tendencias en todos los campos relacionados con las categorías mencionadas anteriormente.



# *Tecnologías y conceptos que debes conocer*

Es muy probable que, llegado a este punto, ya te estés planteando seriamente el convertirte en un ingeniero DevOps. La buena noticia es que seguramente ya cuentas con una buena parte de los conocimientos necesarios para dar ese salto. Sin embargo, hay algunas tecnologías que tienen especial aplicación en este modelo, y si vienes de un paradigma anterior, es posible que no las conozcas. Y lo mismo ocurre con algunos conceptos o, para ser más exactos, metodologías y filosofías de trabajo.

¿Has escuchado alguna vez hablar, por ejemplo, de los microservicios? Si no es así, vas a descubrir que, en una estrategia de segmentación de los entregables, son una herramienta prácticamente indispensable.

Y lo mismo ocurre, puesto que están directamente relacionados, con los contenedores. Una tecnología que permite aislar el software del entorno de ejecución y que facilita aprovisionamiento y despliegues de una manera nunca vista hasta el momento.

También hablaremos de orquestación, repositorios basados en Git, herramientas para automatizar diversas tareas, desarrollo y entrega continuos. Como puedes ver, hay mucho que aprender si deseas tener un perfil de ingeniero DevOps. La buena noticia es que es una vez que los conozcas, ya estarás en disposición de abordar un proyecto en el que se emplee esta metodología.



## Microservicios

El auge de DevOps y la arquitectura de los servicios van de la mano. Y es que, ante la necesidad definida por la metodología ágil de dividir los desarrollos en una estructura de servicios interconectados entre sí, los microservicios son una solución perfecta.

Como su propio nombre indica, hablamos de procesos independientes, que funcionan de manera autónoma y que, por lo tanto, pueden ser gestionados de manera individual. Esto, como ya habrás imaginado, permite que sean desplegados, modificados, etcétera, de una manera muchísimo más ligera que si fueran parte del código de un desarrollo mastodóntico.

Pero claro, un montón de microservicios funcionando de manera autónoma no servirían de nada, si no fuera por su integración. Para tal fin, lo más común es emplear mecanismos singularmente ligeros, generalmente sobre http, aunque también se pueden emplear otros protocolos como AMPQ para conexiones asíncronas.

En cualquier caso, el empleo de conexiones livianas permite establecer muchos enlaces entre los múltiples elementos que componen el desarrollo, sin que esto penalice en modo alguno su rendimiento y fiabilidad.

Así, a la hora de realizar la planificación inicial del proyecto, tras la toma de requisitos, la clave es dividir las funciones en tantos elementos autónomos (que luego se interconectarán) como sea posible.

Y aquí llega otra de las grandes virtudes de los microservicios: una vez definidos los que serán necesarios, podremos elegir en qué entorno queremos desarrollar cada uno de ellos. Efectivamente, podemos elegir el entorno y

lenguaje más adecuado para desarrollar cada uno de ellos. Esta flexibilidad nos permite afinar al máximo, puesto que podremos, si tenemos una base sólida de conocimientos de programación en varios lenguajes, sacar partido a las singularidades de cada uno de ellos.

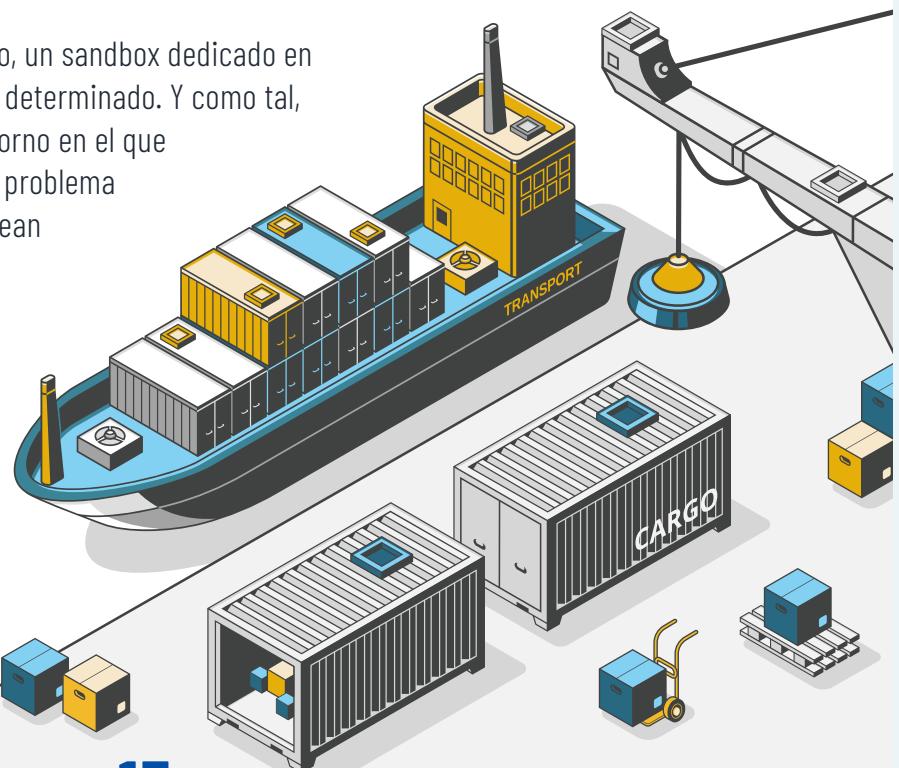
Con este modelo, gracias a su modularidad, es posible realizar rápidamente cambios en cualquier componente del entramado, sin que ello afecte al resto, lo que representa un nivel de agilidad impensable en el paradigma clásico.

## Contenedores

Algunas de las dificultades que nos solemos encontrar al plantear la fase de despliegue son la necesidad de entornos de ejecución muy diversos, desconocimiento de la demanda real a la que se van a ver sometidos algunos procesos, cálculos extremadamente complejos para un aprovisionamiento de recursos adecuado... La respuesta a todos estos problemas nos la ofrecen los contenedores.

Definidos de una manera sencilla, son entornos de ejecución virtualizados. Pero ojo, esto no debe hacer que los confundamos con máquinas virtuales, ya que a diferencia de éstas, la escalabilidad tanto horizontal como vertical es mucho mayor. Tanto es así que, en un entorno DevOps, lo común es emplear múltiples contenedores en varias máquinas virtuales, repartidas entre uno o varios servidores.

El contenedor es, en su funcionamiento, un sandbox dedicado en exclusiva a la ejecución de un proceso determinado. Y como tal, su funcionamiento no depende del entorno en el que se ejecuta. De esta manera, se evita el problema de depender de varios servidores (ya sean reales o virtualizados) en los que crear los distintos ambientes que pueden necesitar los procesos para funcionar correctamente.

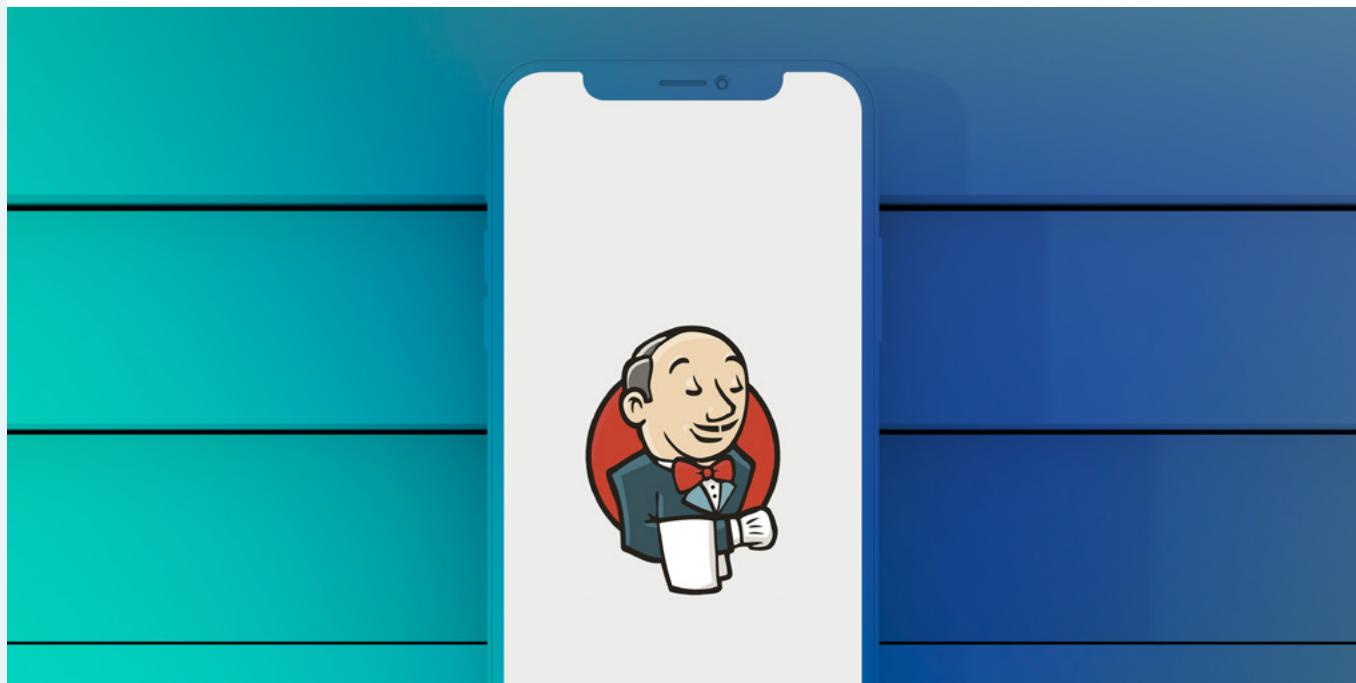


Además, su aislamiento es una garantía de seguridad. Y es que, incluso en el hipotético caso de que un atacante lograra llegar hasta el sandbox en el que está corriendo un servicio, solo tendría acceso al mismo. De esta manera se limita el potencial dañino del ataque, además de facilitar la detección y posterior toma de medidas en caso de que la seguridad se haya visto comprometida.

En cuanto al aprovisionamiento, se ha mencionado antes la escalabilidad horizontal y vertical. Y es que es posible tanto asignar más recursos a un contenedor (vertical) como añadir nuevos contenedores (horizontal) que, mediante un orquestador o un sistema de balanceo de carga, redistribuya la carga de trabajo entre todos los nodos disponibles. El despliegue de un nuevo contenedor, a partir de una plantilla predefinida, es un proceso muy rápido y enormemente sencillo.

## Automatizadores

La eficiencia y la agilidad son elementos clave del desarrollo basado en el modelo DevOps. Y la manera más inmediata de avanzar en ese sentido es, sin duda, la automatización de tareas y procesos. Y es que bajo el modelo tradicional de desarrollo, todavía se siguen efectuando de manera manual muchas tareas que, gracias a herramientas diseñadas para tal fin, pueden ser gestionadas de manera desatendida.



¿Y qué se puede automatizar? Pues con soluciones como Jenkins (sin duda la más popular), gracias a su enorme modularidad y, sobre todo, a la enorme colección de plugins con la que cuenta, es posible conectarlo a la inmensa mayoría de las herramientas que se encuentran habitualmente en las toolchains de DevOps.

Una herramienta de este tipo, una vez configurada en nuestro entorno, nos permitirá crear y controlar versiones, realizar compilaciones de código, llevar a cabo pruebas definidas previamente, gestionar el despliegue... Es cierto que habrá que dedicar un tiempo a ajustar su funcionamiento a nuestras necesidades, pero una vez hecho esto, lo recuperaremos rápidamente (y con creces) a medida que vaya desplegando todas sus funciones.

## Orquestación

Lo más probable es que, a medida que avancemos en el diseño del proyecto, detectemos que es necesario emplear distintos tipos de recursos: servidores, bases de datos, máquinas virtualizadas, dominios, redes virtuales y un largo etcétera.

Gestionar un ecosistema de este tipo puede resultar un tanto complejo, puesto que son muchos los elementos que lo forman, y hay que prestar también especial atención a los enlaces que los interconectan.

Un orquestador es, como su propio nombre ya da a entender, la herramienta que emplearemos para gestionar, de manera centralizada, todos los recursos empleados en la infraestructura IT de nuestro proyecto.

Con el orquestador, además de desplegar la infraestructura, también tendremos herramientas para monitorizar su estado y funcionamiento y, lo que es también crucial en DevOps, podremos automatizar múltiples tareas (algunas bastante tediosas y repetitivas) que, de otro modo, pueden convertirse en un sumidero de recursos.





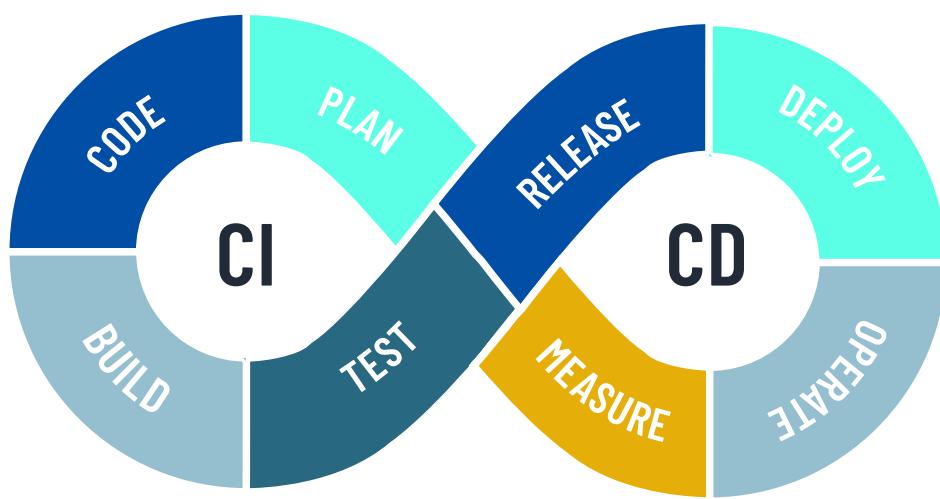
## Git

Cualquier profesional del desarrollo de software, y especialmente los responsables de proyectos, saben que gestionar de manera adecuada el control de versiones puede ser un auténtico suplicio. Consciente de ello, y de la necesidad de una solución que permitiera administrar adecuadamente versiones de proyectos compuestos por un gran volumen de fuentes, Linus Torvalds diseñó este sistema.

Entre los factores que lo hacen integrarse tan bien dentro de la filosofía DevOps, destaca sin duda el hecho de que fue diseñado pensando en desarrollo no lineal y organizado en ramas (branches). De esta manera la estructura de almacenamiento de versiones siempre se ajustará a los distintos ritmos de los componentes del proyecto, aunque sus ciclos sean distintos.

También destaca por ser una herramienta diseñada para ser eficiente en proyectos grandes. Esto no quiere decir que no sea útil para proyectos más modestos, sino que gracias a las múltiples herramientas de comparación de versiones y otras mejoras de optimización, las revisiones serán mucho más rápidas y efectivas que con otros sistemas de control de versiones.

Como ventaja adicional, al conservar todo el histórico de versiones, permite realizar rápidamente un rollback si detectamos cualquier problema en la versión actual.



## ***Integración y entrega continuos (CI/CD)***

CI/CD es otro elemento clave de DevOps, hasta el punto de que, probablemente, de manera individual es el que mejor sirve para definir la filosofía global de este paradigma. De manera resumida, la propuesta de este modelo consiste en agilizar el proceso de modificación de código y publicación de actualizaciones.

Como en otros muchos elementos de DevOps, en CI/CD se saca partido de las funciones de automatización en la publicación (entrega) de nuevas versiones. Para ello se apoya, principalmente, en dos pilares: el empleo de un Git para el control y la gestión de versiones, ya que el “corto” periodo de vida de cada actualización, empuja a los equipos a desarrollar cambios sencillos de manera continua.

En determinadas circunstancias se puede alargar la vigencia de una versión. Por ejemplo, si es necesario realizar cambios mayores en la misma. Esto no es incompatible con CI/CD, siempre y cuando una vez realizado este cambio, se recupera la frecuencia de actualizaciones.

# Producto mínimo viable

Por norma general, la búsqueda de la excelencia en el desarrollo de proyectos tiene un efecto perverso: la búsqueda de la perfección. Pretender que la versión 1.0 incluya el 100% de las funcionalidades que se desean se traducirá, con toda probabilidad, en enormes tiempos desde la toma de requisitos hasta la puesta en marcha del software.

Para evitar este problema, el enfoque más adecuado es optar por el modelo de producto mínimo viable, que no es otra cosa que una definición de qué elementos son los necesarios para satisfacer la demanda inicial. Esto requerirá de un feedback previo al inicio del desarrollo, en el que tendremos en cuenta, claro, todos los requisitos del proyecto, pero priorizaremos los más urgentes y, en base a los mismos, definiremos una versión inicial.

De este modo reduciremos el tiempo de espera entre el inicio del proyecto y la puesta en marcha del desarrollo. Y, a partir de ese punto, siguiendo la filosofía CI/CD, iremos completando la lista de requisitos en entregables que serán desplegados con bastante frecuencia.

Otra ventaja que nos ofrece la publicación, en primera instancia, del producto mínimo viable, es que empezaremos a recibir feedback sobre el mismo de manera temprana. De esta manera, si se detectan problemas con la concepción inicial del proyecto que es necesario corregir, podremos evitar muchas horas de trabajo perdido.



# #PreguntasFrecuentes

*¿Es lo mismo Lean y Agile?  
¿Son compatibles?*

No, DevOps no es sinónimo de ambos conceptos, aunque sí que es cierto que tienen una relación muy directa. Lean propone un completo sistema de organización de todas las actividades de una empresa. Para ello busca reducir las pérdidas de recursos y maximizar la entrega final de valor al cliente. Aunque tiene su origen en el sector industrial y de manufactura, es aplicable a prácticamente cualquier tipo de empresa.

Agile, por su parte, es una traslación de Lean al desarrollo de software. Así, toma los principios de su filosofía y los adapta a las particularidades de la gestión de proyectos de desarrollo. Y en cuanto a DevOps, se trata de una metodología de trabajo que nace y se desarrolla a partir de los conceptos previamente definidos por Agile.

## *¿Qué programa / solución necesito para desarrollar con DevOps?*

Si todavía te haces esta pregunta, lo más probable es que hayas saltado directamente a la misma desde el índice. Y es que, como ya te hemos contado, no hay una herramienta (o un entorno) que cubra todas las necesidades que plantea un proyecto desarrollado siguiendo la filosofía DevOps.

Lo que sí que puede resultarte muy interesantes, si estás buscando herramientas para acometer un proyecto, es localizar las toolchains más populares. Estas cadenas de herramientas, agrupadas para cada una de las tareas en las que se divide un proyecto, te ofrecen una manera rápida y fiable de equiparte con todo lo que puedes necesitar.

# *¿Tiene aplicación DevOps en IoT? ¿Y en cloud?*

Wikipedia data el origen de DevOps en 2008, en una conferencia sobre Agile. Es decir, que se trata de un paradigma bastante actual y, por lo tanto, desarrollado teniendo en cuenta estas tecnologías que, si bien en ese momento todavía eran emergentes, a día de hoy se han popularizado incluso más de lo que cabía esperar entonces.

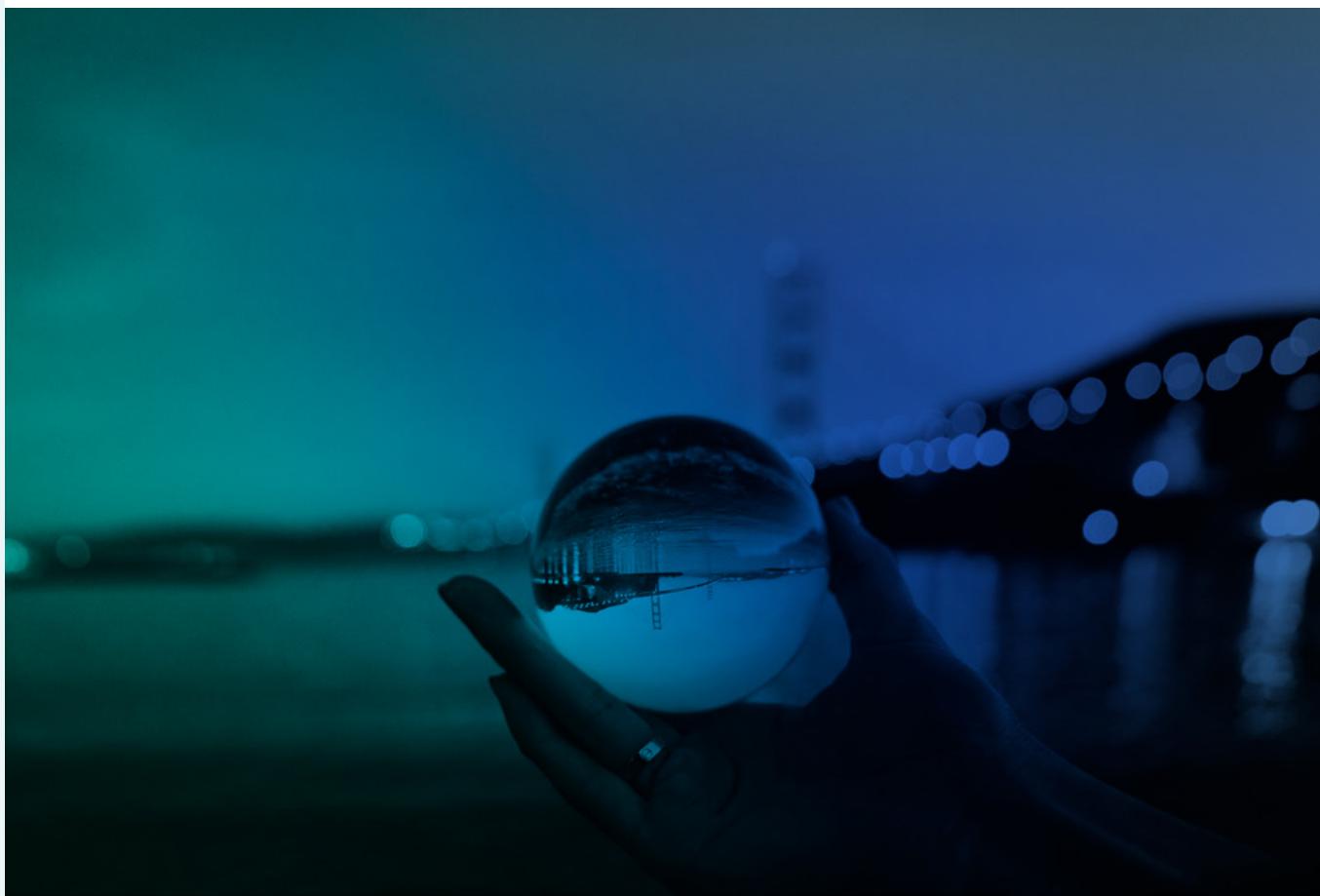
Si entendemos los diversos componentes IoT de una infraestructura como esos elementos independientes de los que hablábamos en la definición de un proyecto bajo DevOps, veremos que no solo tiene aplicación, sino que es, probablemente, la metodología más adecuada para un proyecto en el que la entrada (y en según qué casos también la salida) de múltiples elementos puede ser una constante.

Y con respecto a cloud no es que tenga aplicación, es que es la filosofía misma de los escenarios de nube híbrida. Casi se podría decir que, en muchos casos, el proyecto realizado bajo DevOps y una infraestructura de nube híbrida no son sino dos caras de una misma moneda.

# *¿Puedo transicionar un proyecto estándar a DevOps?*

La respuesta muy corta es "Sí". Ahora bien, la respuesta un poquito menos corta es "Sí, pero habría que verlo". En muchas ocasiones, partiremos de un gran desarrollo ya efectuado con anterioridad, y es posible que deseemos "migrar" su gestión a un ambiente DevOps. En tal caso, solo si tenemos un conocimiento muy sólido de la solución legacy, puede tener sentido que vayamos añadiendo nuevos elementos a la misma en forma de microservicios, contenedores, etcétera. A este fin, sistemas como Kubernetes nos pueden resultar de gran utilidad.

Sin embargo, en todos los casos, no es recomendable mantener más allá de lo imprescindible un desarrollo legacy. Por lo tanto, lo que tendremos que ir haciendo, mientras aún lo mantenemos en funcionamiento, es ir diseñando y desplegando los nuevos elementos del desarrollo destinados a sustituir, progresivamente, las funciones del anterior.



# Más allá de DevOps

DevOps es, no cabe duda, el presente del desarrollo de proyectos. Sin embargo, ¿qué tipo de profesionales seríamos si no estuviéramos observando constantemente para intentar saber qué será lo siguiente?

La buena noticia, si has decidido dar los pasos necesarios para convertirte en un ingeniero DevOps, son que a día de hoy todo apunta a que, aunque con la lógica evolución, la base seguirá siendo la misma durante no pocos años. Es probable que la aparición de nuevas herramientas aporte cambios y novedades al paradigma, y que se definan nuevos estándares que sean adoptados de manera masiva por la comunidad de desarrolladores.

Algo que ya hemos empezado a ver, y que seguramente se agudizará en el futuro, es la inclusión de un nuevo elemento entre Dev y Ops. El primero, y más fuerte hasta el momento, es DevSecOps y sí, como ya habrás deducido, Sec hace referencia a Security. En este caso concreto, se pone en el foco en la aplicación de políticas de seguridad desde el inicio del desarrollo, y la incorporación de las mismas en todo su ciclo de vida.

También veremos, si no a corto sí a medio plazo, la irrupción masiva de las herramientas de inteligencia artificial en los entornos DevOps. A día de hoy, ya lo hemos mencionado, hay bastantes tareas que pueden ser

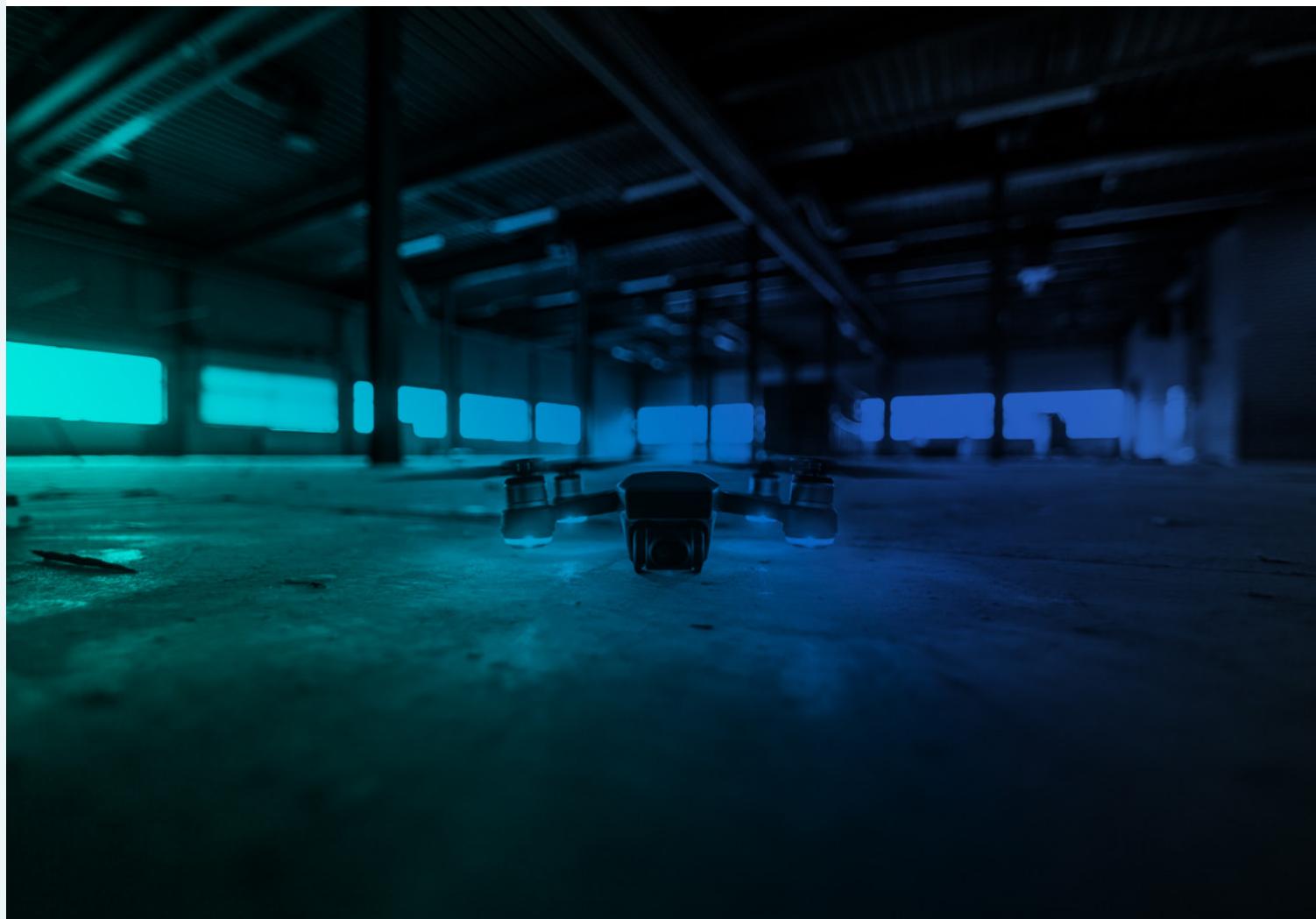
automatizadas, pero la aportación que puede llegar a hacer la IA en este campo es, sencillamente, espectacular.

No parece probable, pero tampoco es imposible que, en algún momento, aparezca un nuevo paradigma que, por sus ventajas, amenace la actual posición destacada de DevOps. Sea por una organización más práctica de las tareas, por dar con un modelo más simple o por cualquier otra causa, no debemos descartar por completo esa posibilidad.

No obstante, si se diera ese caso hay que tener en cuenta algunos factores importantes. El primero es el tiempo que transcurre desde la aparición de un nuevo

modelo hasta su implantación. Y es que, aunque ya llevamos algunos años viviendo una gran proliferación de DevOps, no debemos olvidar que su nacimiento tuvo lugar hace nada menos que doce años.

Además, en el caso de que se diera esta circunstancia, lo más probable es que el nuevo modelo también se base en Agile. Y, por lo tanto, no estaremos hablando de un cambio "rupturista". En su lugar, lo más probable es que tome muchos elementos de DevOps y los aplique a una fórmula mejorada. Así, por lo tanto, muchos de los conocimientos y la experiencia que hayamos adquirido gestionando proyectos en DevOps, serán trasladables al ese nuevo modelo que está por venir.





# DEVOPS

*Tecnología, herramientas  
y plan de carrera*

*Una Guía de MCPRO*

Redacción: David Salces

Diseño y maquetación: Marila Bautista

**ipnet. MCPRO CIONET**