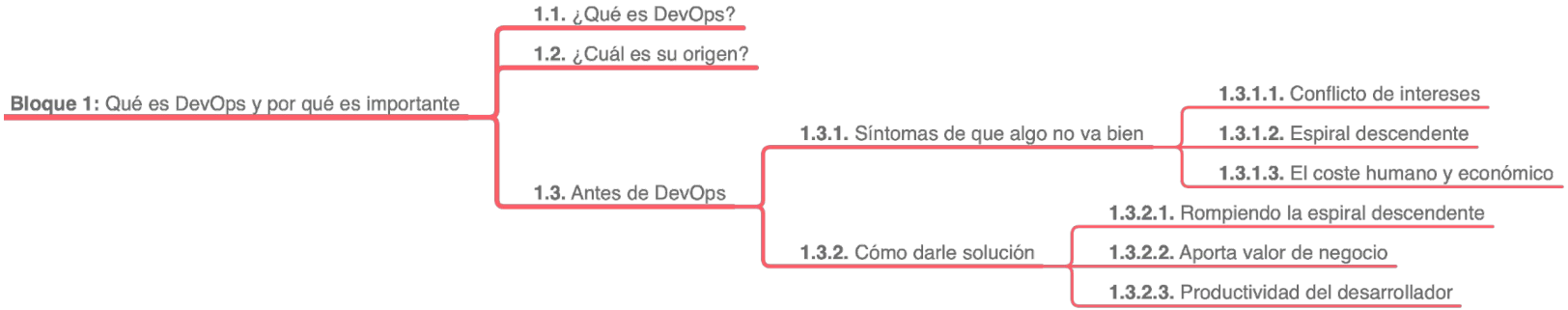
Bloque 1

Qué es DevOps y por qué es importante



~ 1.1 ~

¿Qué es DevOps?

DevOps no solo es una perfil, es una cultura

Personas + Tecnología

Cultura

Qué es DevOps

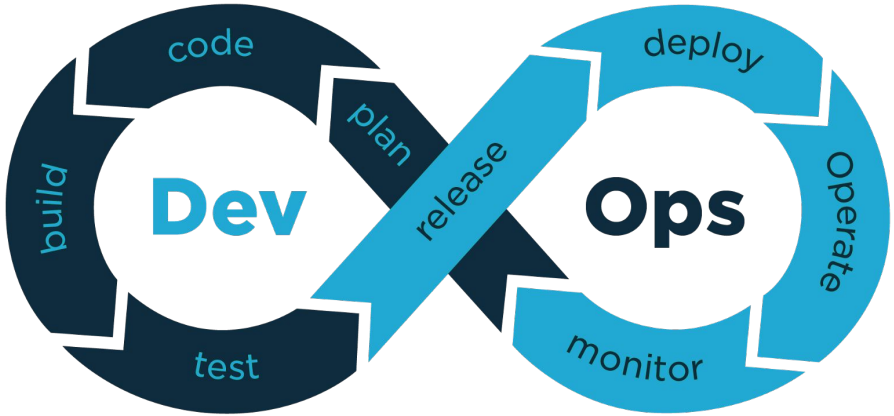
DevOps es una cultura de colaboración y aprendizaje cuyo propósito es acelerar el ciclo de vida del desarrollo de software, desde su ideación hasta su puesta en producción.

Qué es DevOps

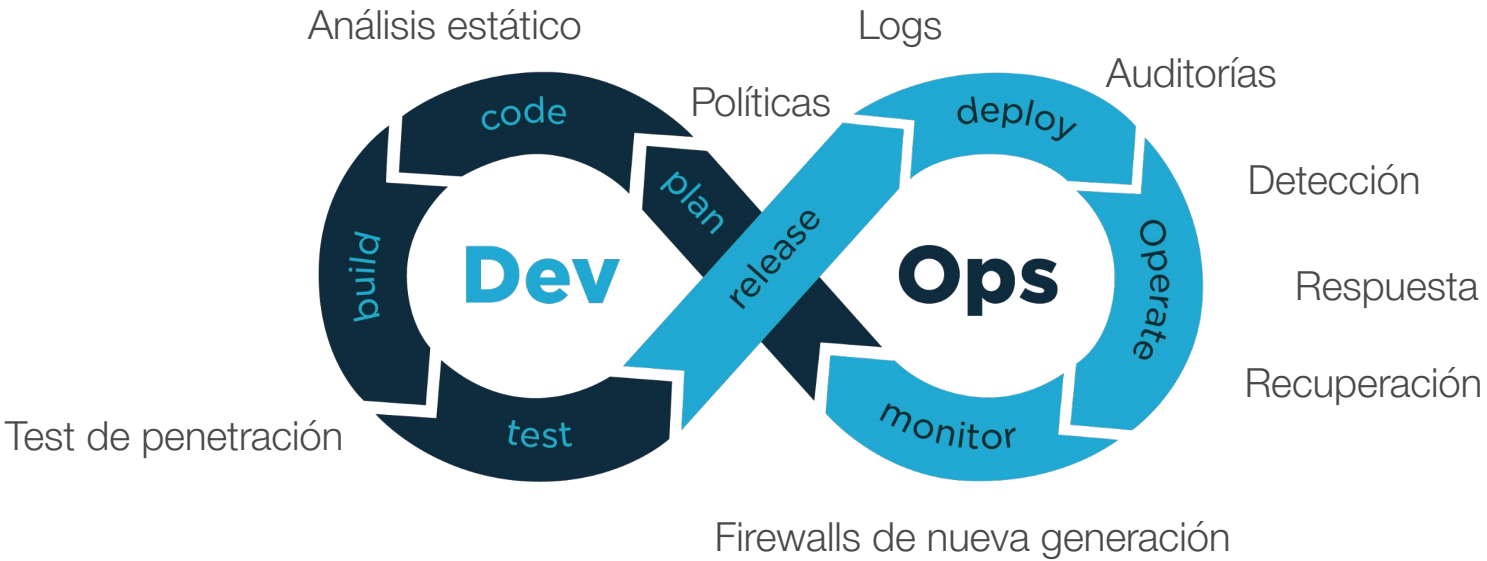
Una cultura **DevOps** ayuda a:

- Reducir la fricción interpersonal
- Eliminar cuellos de botella en los procesos
- Mejorar la colaboración entre compañeros y equipos
- Aumentar la satisfacción personal en el trabajo
- Acelerar la productividad del equipo

Qué es DevOps



Y qué es DevSecOps



~ 1.2 ~

¿Cuál es su origen?

Origen

En 2001 se publicó el Manifesto for Agile Software Development, que recoge 12 principios como respuesta a la frustración e inflexibilidad de los equipos que trabajaban en procesos lineales -waterfall-. Agile revolucionó el desarrollo de software en diversos aspectos, pero no consiguió resolver el conflicto entre desarrolladores y operaciones. **DevOps** surgió para atender este problema y darle solución.

Podríamos decir, por tanto, que **DevOps** es una evolución de Agile.

Origen: Agile Manifesto

Individuos e interacciones	***sobre*** procesos y herramientas
Software funcionando	***sobre*** documentación extensiva
Colaboración con el cliente	***sobre*** negociación contractual
Respuesta ante el cambio	***sobre*** seguir un plan

Site: Agile Manifesto | **Post:** Manifiesto por el Desarrollo Ágil de Software | **Enlace:** http://bit.ly/2J1V5ei

Origen: Principios del Agile Manifesto

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

Origen: Principios del Agile Manifesto

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. El software funcionando es la medida principal de progreso.

8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

Origen: Principios del Agile Manifesto

9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Origen: Lean manufacturing

DevOps toma de Lean manufacturing las ideas de mejora contínua y eliminación de desperdicios -waste-.

La forma más simple de describir los desperdicios es como "**Algo que no aporta valor**".

Transporte / Inventario / Movimientos / Defectos

Tiempos de espera / Sobreprocesamiento / Sobreproducción

~ 1.3 ~

Antes de DevOps

~ 1.3.1 ~

Síntomas de que algo no va bien

Síntomas

Muchas organizaciones son incapaces de hacer cambios en producción en cuestión de minutos y/o desplegar cientos de ellos a lo largo del día. Estos casos suelen llevar semanas o incluso meses, y al no ser una rutina, suele resultar en caídas del sistema y en la necesidad de hacer heroicidades para salvar los despliegues.

Síntomas

Ser competitivo requiere un rápido time-to-market, servicios de alto nivel y continua experimentación. Estas organizaciones están en clara desventaja competitiva, en gran parte porque son incapaces de resolver su principal problema: conflicto de intereses, el cual se explica a continuación.

Síntomas

Time-to-market (TTM)

Hace referencia al periodo de tiempo que toma un producto desde que es concebido hasta que está disponible para su venta.

~ 1.3.1.1 ~

Conflicto de intereses

Conflicto de intereses

El conflicto entre **Dev** y **Ops** viene motivado por el hecho de que estos equipos tienen objetivos que compiten entre ellos. Esto se traduce en un time-to-market lento, una calidad del producto cuestionable y un aumento de la deuda técnica.

Una organización de IT tiene entre sus objetivos:

- Responder a un mercado que cambia continuamente.
- Proveer de servicios estables, fiables y seguros a sus clientes.

Conflicto de intereses

Dev tiene que responder a los cambios del mercado, llevando nuevas funcionalidades a producción de la manera más rápida posible.

Ops tiene la responsabilidad de ofrecer servicios estables, fiables y seguros al cliente, lo que dificulta llevar a producción cambios que puedan poner en peligro el entorno.

Conflicto de intereses

Se forman, por tanto, silos de responsabilidades:

- **Dev** no sabe cómo liberar y dar soporte a su propio código, ya que desconocen los sistemas y sus requerimientos. Muchos desconocen cómo se ejecuta su código, o incluso no les importa.
- **Ops** se ocupa del despliegue de código y se asegura de que corra perfectamente. A veces asumen la culpa de problemas derivados de un código mal escrito.

Conflicto de intereses

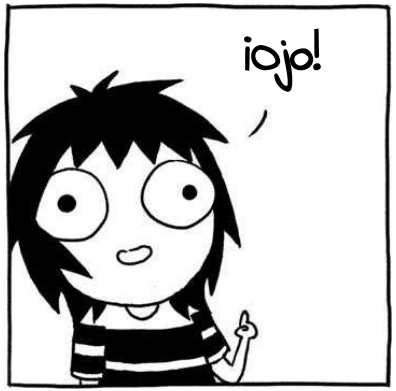
> *Esta situación se da cuando las medidas que se toman a lo largo de los diferentes silos impiden lograr los objetivos globales de la organización.*

El Dr. Eliyahu M. Goldratt llamó a esto *the core, chronic conflict*.

~ 1.3.1.2 ~

Espiral descendente

Espiral descendente



Deuda técnica

Describe cómo las decisiones que tomamos nos llevan a problemas que, con el tiempo, complican cada vez más su resolución, y reducen las opciones disponibles para ello en el futuro.

Tiempo VS Calidad

Para ello se recomienda usar un 20% del tiempo en solventar este problema.

Espiral descendente

Sucede en tres fases:

- A veces, **Ops** debe mantener aplicaciones e infraestructuras complejas, frágiles y mal documentadas. Cualquier cambio, por pequeño que sea, pone en peligro estos sistemas.
- **Dev** puede verse obligado a cumplir con exigencias y/o urgencias adquiridas por compromisos de otros. Esto supone un aumento en la deuda técnica al tener que entregar en fecha.

Espiral descendente

- Por último, se desencadenan una serie de complicaciones: las personas están cada vez más ocupadas, las tareas toman más tiempo del debido, las comunicaciones se vuelven algo más lentas.

Espiral descendente

Como resultado de lo anterior:

- Trabajo acoplado: pequeñas tareas suponen grandes problemas.
- Escepticismo y poca tolerancia sobre asumir nuevos cambios.
- Dependencias que generan tiempos de espera no deseados.

~ 1.3.1.3 ~

El coste humano y económico

Coste

Cuando las personas están atrapadas en una espiral descendente durante un tiempo prolongado sucede lo siguiente:

- Aparece el agotamiento, la fatiga, el cinismo y la desesperación como resultado de trabajar en un sistema problemático que no admite propuestas de cambios y mejoras.
- Cultura movida por el miedo a actuar correctamente y recibir castigo, fallar o ver peligrar el puesto de trabajo.

Coste

El resultado es una situación de indefensión aprendida, en la que las personas se vuelven reacias o incapaces de actuar de forma que se evite y prevenga el problema a futuro.

También afecta a nivel económico: se pierde mucho capital en trabajos no planeados, urgencias y retrabajos. **DevOps** ayuda a reducir esta pérdida y redirigir el potencial humano hacia tareas que aporten más valor.

~ Resumen 1.3.1 ~

Dev y **Ops** persiguen objetivos distintos.

Decisiones y procedimientos que llevan al desorden y al caos.

Indefensión aprendida: nula voluntad a poner remedio.

Incremento de coste por culpa de urgencias y retrabajos.

~ 1.3.2 ~

Cómo darle solución

~ 1.3.2.1 ~

Rompiendo la espiral descendente

Rompiendo la espiral

Las siguientes medidas pueden ayudar a prevenir la espiral:

- Trabajar en equipos pequeños autosuficientes que puedan implementar nuevas funcionalidades, validarlas y desplegarlas en producción de manera rápida y segura. De esta forma hacemos de los despliegues algo rutinario y predecible.

Rompiendo la espiral

- Crear ciclos de feedback en cada paso del proceso, de manera que cualquiera puede ver de inmediato los efectos de sus acciones.
- Automatizar la ejecución de tests ayuda a los desarrolladores a descubrir errores rápidamente y actuar en el momento, en vez de generar deuda técnica. Además, garantiza que el código opera correctamente para lo que fue escrito.

Rompiendo la espiral

- Obtener métricas, tanto de nuestro código como del entorno de producción, asegura que los problemas se detectan y solucionan rápidamente.
- Generar conocimiento de cualquier error encontrado, y así poder prevenir el caer de nuevo en él o en otros similares.

Rompiendo la espiral

- Crear una cultura de confianza y colaboración: nos apoyamos en tests y peer reviews para ganar confianza en que los problemas no impacten al cliente.
- Realizar post-mortems para entender mejor la raíz del problema y así poder prevenirlo en el futuro; esto enriquece la cultura de aprendizaje.

Rompiendo la espiral

- Introducir fallos en producción de manera controlada para aprender cómo y dónde el sistema pone de manifiesto sus puntos débiles. Esto ayuda a mejorar la resiliencia.
- Transmitir un sentimiento de propiedad del trabajo. Todos confiamos en que nuestro trabajo importa y contribuye de manera significativa a lograr los objetivos de la organización.

Rompiendo la espiral

Peer review

En desarrollo de software, es una revisión de código llevada a cabo por su autor y por uno o varios de sus compañeros, con el objetivo de evaluar su calidad.

Rompiendo la espiral

Post-mortem

Es un proceso para determinar y analizar qué elementos de un proyecto fueron satisfactorios y que otros no lo fueron. Se suele hacer al finalizar el proyecto o uno de sus ciclos de desarrollo.

~ 1.3.2.2 ~

Aporta valor de negocio

Valor de negocio

Puppet Labs realiza un informe anual para entender en profundidad la salud y hábitos de las organizaciones en cada fase de adopción de **DevOps**. Estas son algunas conclusiones que pueden extraerse de los datos que figuran en los informes de 2013 a 2016:

- Se obtienen evidencias empíricas de que **DevOps** acaba con el problema de conflicto de intereses, comentado anteriormente.

Valor de negocio

- Las organizaciones de alto rendimiento tuvieron en dicho periodo el doble de probabilidad de obtener rentabilidad, cuota de mercado y objetivos de productividad.
- La satisfacción laboral de los empleados creció, cesando la tasa de abandono y subiendo la identidad corporativa.
- Al añadir objetivos de seguridad en todas las fases del proceso de desarrollo, se redujo un 50% el tiempo dedicado solucionar problemas de este tipo.

~ 1.3.2.3 ~

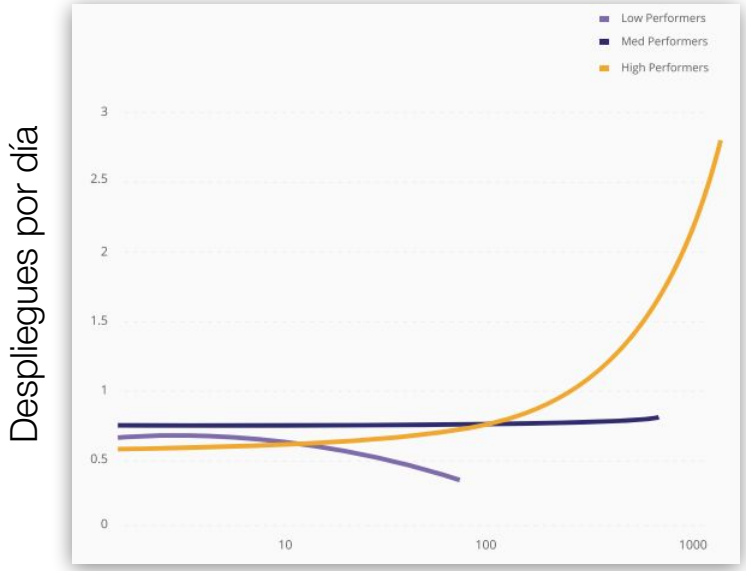
Productividad del desarrollador

Productividad

Cuando el número de desarrolladores crece, la productividad se ve gravemente afectada por temas de comunicación, integración, testing, etc. Y no sólo la individual, sino la de toda la organización.

DevOps demuestra que teniendo la arquitectura correcta, siguiendo las buenas prácticas y las normas culturales apropiadas, los equipos pueden desarrollar, integrar, testar y desplegar de manera independiente, segura y rápida.

Productividad



Site: Puppet | **Post:** 2015 State of DevOps Report | **Enlace:** http://bit.ly/2Zfx5eg

~ Resumen 1.3.2 ~

Se deben habilitar mecanismos que ayuden a contrarrestar los efectos del caos generado por la espiral descendente.

DevOps bien aplicado mejora la productividad del desarrollador, así como el valor de negocio de la organización.