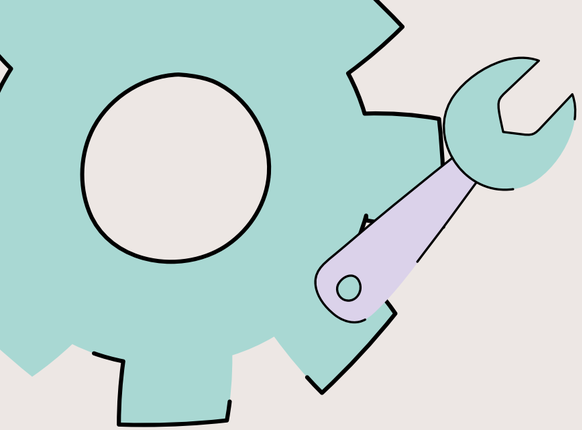
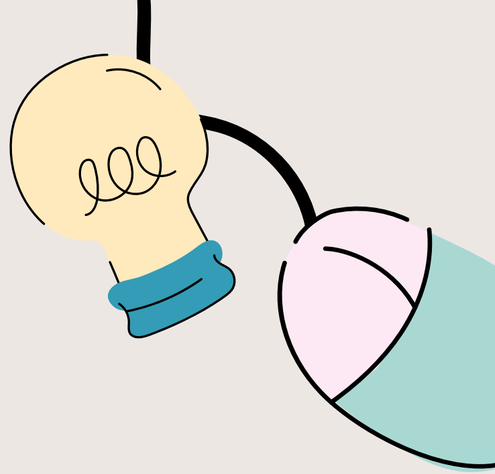


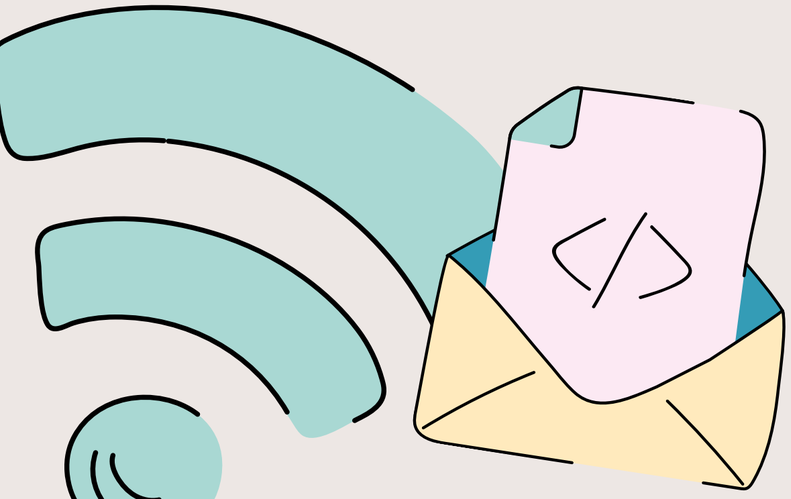


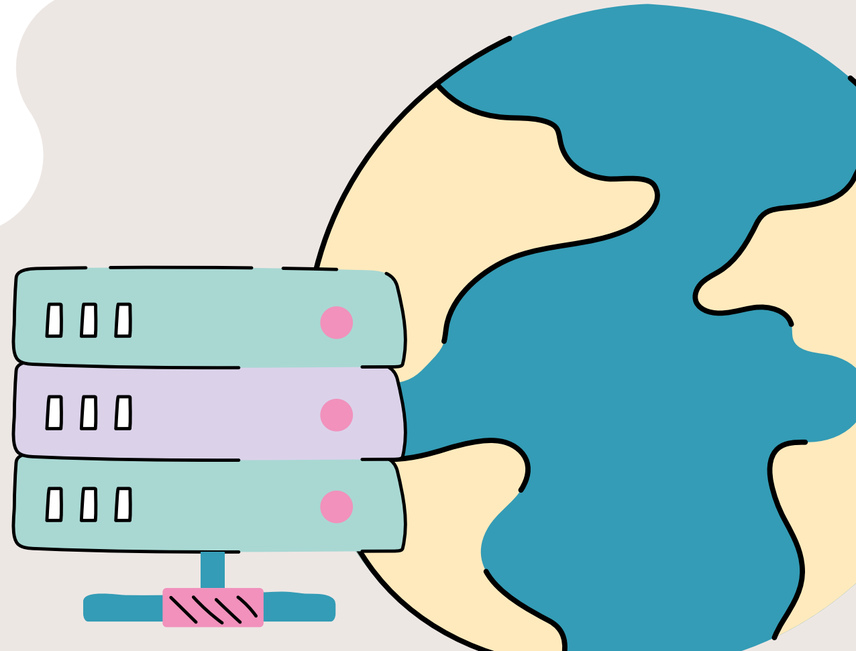
Desarrollo de Software

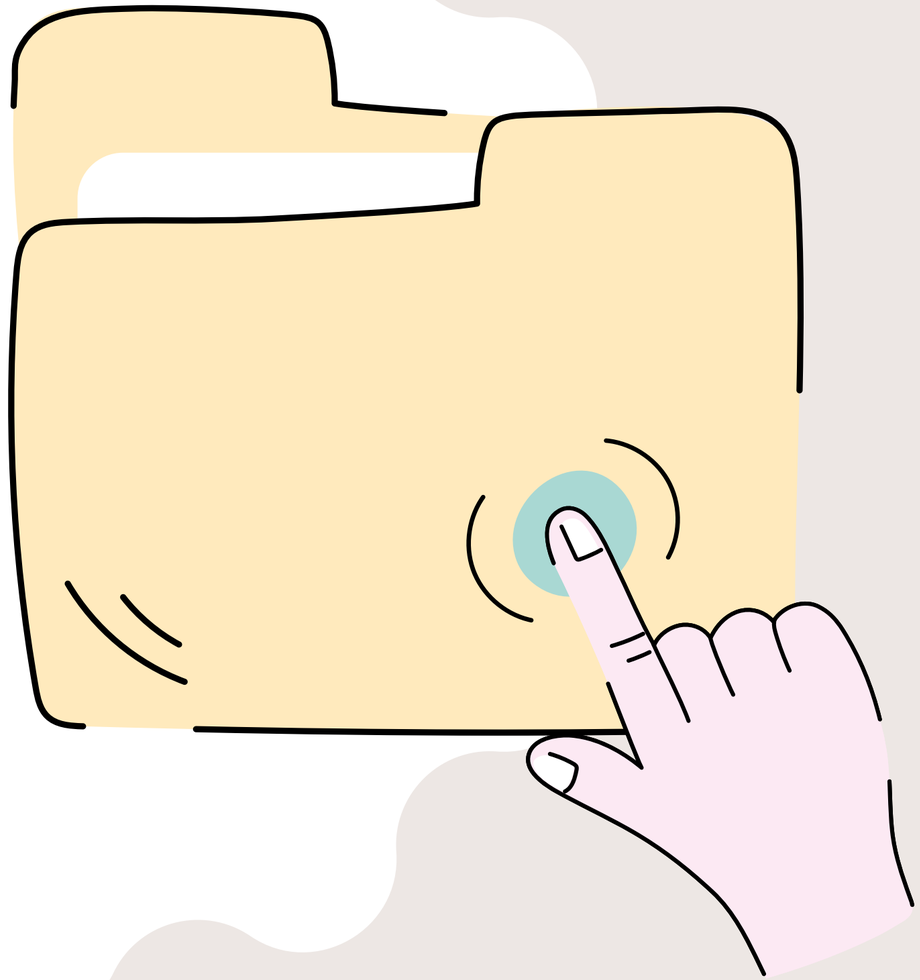
Actividad 1

Aarón Flores Alberca

Diego Delgado Velarde





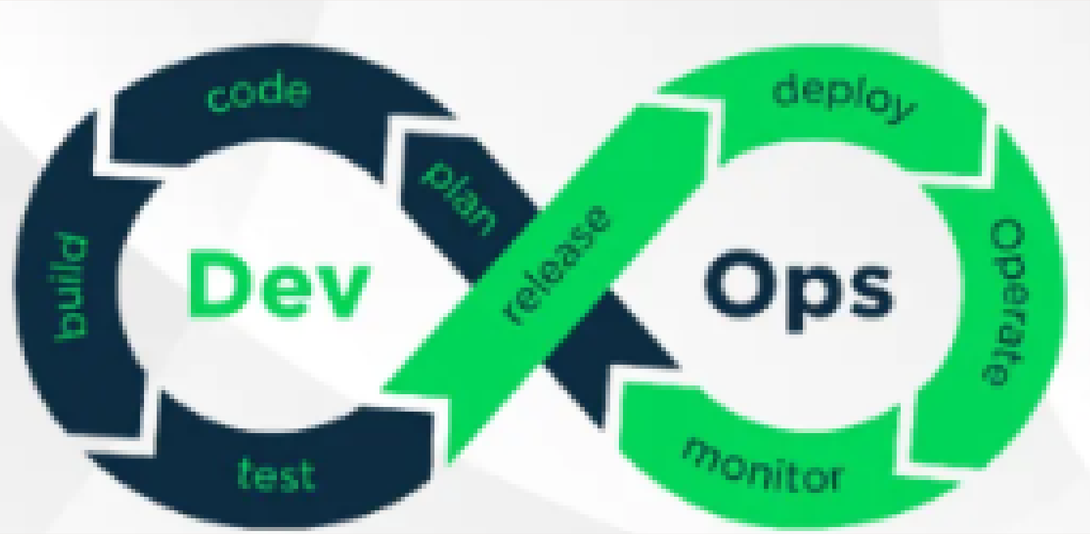


Contenidos

- **Parte 1**
  - Objetivo: Comprender los principios fundamentales de DevOps y diferenciar qué es y qué no es DevOps.
- **Parte 2**
  - Objetivo: Comprender los conceptos clave y la importancia de DevSecOps, Infraestructura como Código (IaC), Observabilidad, Experiencia del desarrollador, InnerSource y Ingeniería de plataformas en el contexto de DevOps.

Parte 1

¿Por qué surgió la necesidad de DevOps en el desarrollo de software?

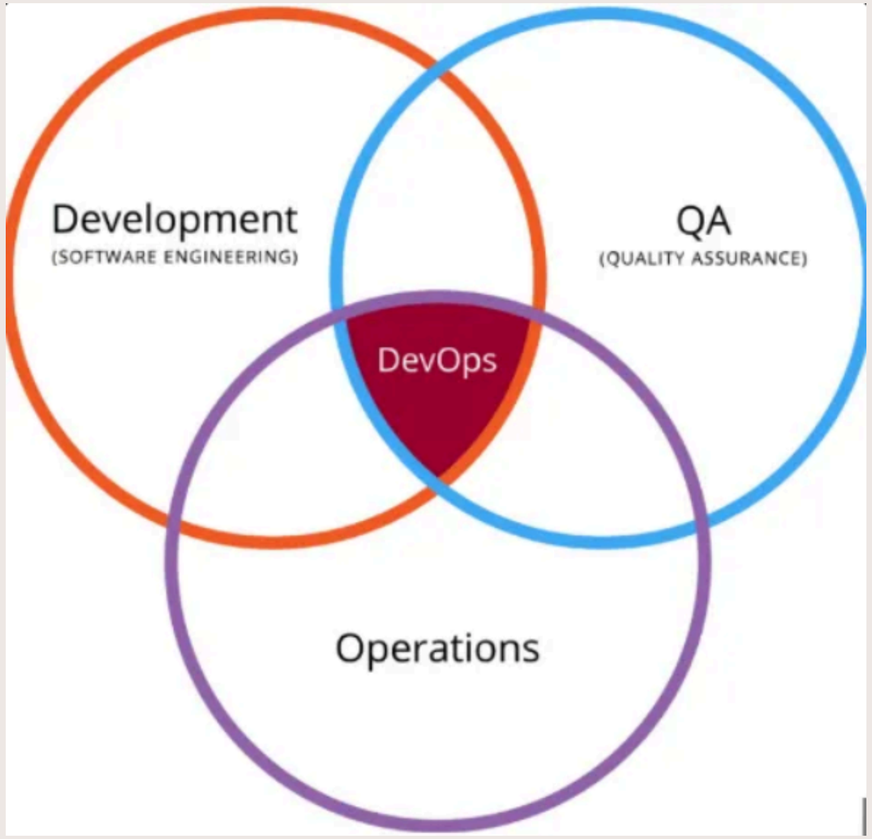


La necesidad de DevOps surgió porque el modelo tradicional de desarrollo generaba retrasos debido a que el equipo de desarrollo y operaciones trabajan de forma aislada, es decir, solo se comunicaban cuando debían pasar el proyecto al siguiente equipo y esto generaba problemas en la entrega de software.

Los ciclos de lanzamiento eran largos, los errores se detectaban tarde, y había conflictos entre los equipos de desarrollo y operaciones debido a sus objetivos contradictorios. Ante ello surge DevOps para fomentar el trabajo colaborativo desde el inicio, acelerar y mejorar la calidad del software.

Parte 1

Explica cómo la falta de comunicación y coordinación entre los equipos de desarrollo y operaciones en el pasado llevó a la creación de DevOps.



En el pasado cada equipo trabajaba de manera aislada y tenía objetivos diferentes, por ejemplo, el equipo de desarrollo quería agregar nuevas funcionalidades y entregar código rápido; mientras el equipo de operaciones evitaba cambios “bruscos”, considerando la estabilidad del sistema como máxima prioridad, todo ello genero una contradicción y una clara falta de coordinación.

Entonces se crea DevOps, para que estos dos grandes equipos colaboren conjuntamente desde el inicio, como una entidad unificada, persiguiendo un mismo objetivo la de acelerar los lanzamientos, mejorar la calidad del producto y, en consecuencia se brinda un excelente servicio a los clientes.

Parte 1

Describe cómo el principio de mejora continua impacta tanto en los aspectos técnicos como en los culturales de una organización.

- A nivel **técnico**: Permite cualidades como la confiabilidad, adaptabilidad y eficiencia de los procesos de entrega de software a través de un análisis continuo de métricas de rendimiento y la utilización de flujos de trabajo automatizados.

- A nivel **cultural**: Fomenta un trabajo colaborativo, asegurando responsabilidad, ciclos de retroalimentación y ayudando a desmantelar los silos organizacionales.

Parte 1

¿Qué significa que DevOps no se trata solo de herramientas, individuos o procesos?

DevOps no se trata solo de herramientas debido a que, a pesar que uno domine Docker, Kubernetes, GitHub Actions u otras herramientas y siga arquitecturas conocidas, no significa que haya logrado DevOps, ya que siempre hay individuos y procesos detrás de la herramienta.

En adición DevOps tampoco se trata solo de individuos ya que dichos individuos necesitan trabajar de manera colaborativa y utilizar herramientas tecnológicas. Además, DevOps no se trata solo de introducir un nuevo proceso en sí mismo, sino es más que una fusión de desarrollo y operaciones, roles específicos, procesos, herramientas, tecnología para superar la fricción creada por los silos.

Parte 1

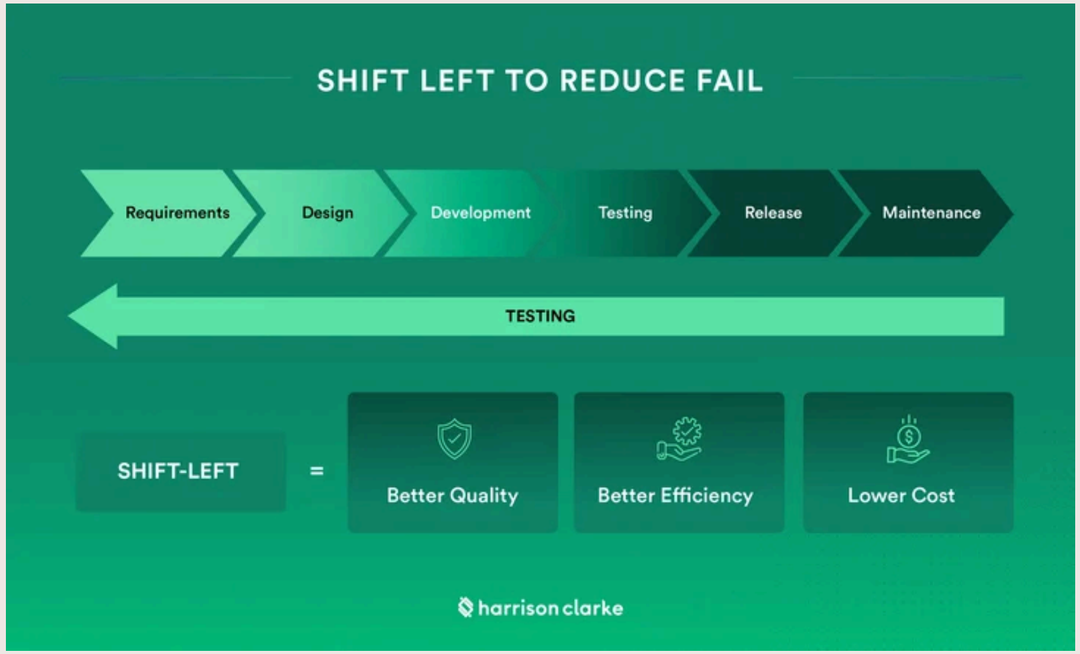
Según el texto, ¿cómo contribuyen los equipos autónomos y multifuncionales a una implementación exitosa de DevOps?

Contribuyen agilizando el flujo de trabajo ya que los equipos trabajan de manera colaborativa desde el inicio hasta la entrega del producto, además dichos equipos poseen una variedad de habilidades, desde la codificación y pruebas hasta la implementación, por ende el equipo no tiene que esperar a departamentos externos para tomar decisiones.

En adición disuelve silos que a menudo causan fricción dentro de las organizaciones, el resultado es un proceso simplificado que facilita tomar decisiones rápidas sin depender de otros equipos, fomenta una cultura de colaboración y responsabilidad, reduce tiempos de entrega, mejora la calidad del software y responde rápidamente a los cambios.

Parte 2

¿Qué significa desplazar a la izquierda en el contexto de DevSecOps y por qué es importante?



Se denomina “desplazamiento a la izquierda” o “shift left” a la integración de prácticas de seguridad desde etapas tempranas del ciclo de desarrollo, lo que va en contra del modelo tradicional del desarrollo de software en el que se aplazaba a practicas de seguridad hasta el final de dicho ciclo.

Un ejemplo claro pudiese ser las de integración de escáneres de vulnerabilidades y análisis de código dentro del pipeline CI/CD.
Este concepto es de suma importancia porque permite la identificación de posibles riesgos desde antes, lo que puede reducir costos de hasta cientos de miles de dólares en etapas tardías del desarrollo; además, que fomenta una cultura de seguridad global más allá de que sea el trabajo de un departamento externo.

¿Cómo IaC mejora la consistencia y escalabilidad en la gestión de infraestructuras?

Se entiende a IaC como la gestión de infraestructura de TI a partir de un proveedor local o de nube mediante código.

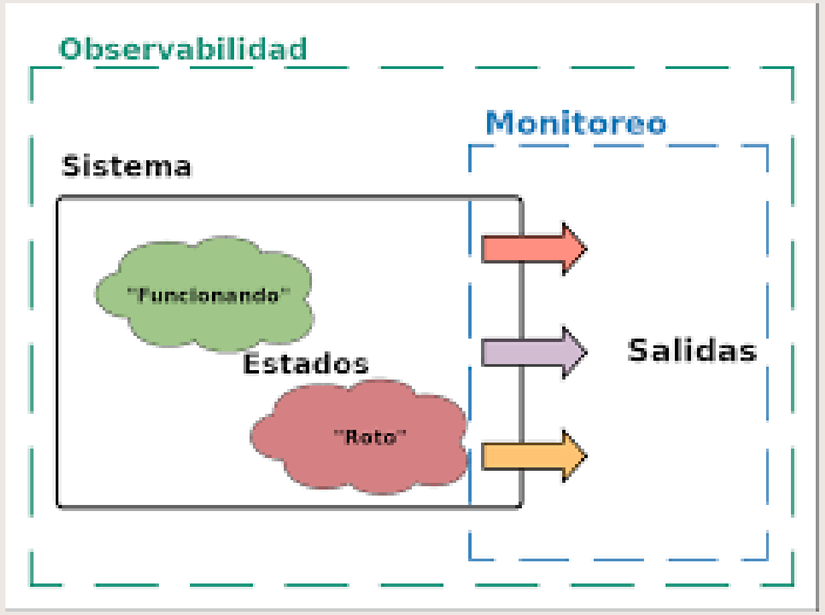
Esta práctica es ventajosa frente a otras alternativas gracias a conceptos clave como:

- La **reproducibilidad** que permiten que cualquier persona o proceso recrear entornos idénticos con el archivo de configuración correcto
- El **control de versiones** que otorga una documentación detallada que facilita la detección de deficiencias y permitiendo rollbacks si es necesario.
- La **composibilidad** que descompone la infraestructura en módulos reutilizables que permiten crear infraestructuras mas complejas.
- La **idempotencia** que asegura que la futura ejecución repetida de las herramientas IaC no genere cambios adicionales establecido con la configuración establecida.

Contamos con ejemplos como AWS Cloud Development Kit, Terraform o Pulumi

¿Cuál es la diferencia entre monitoreo y observabilidad? ¿Por qué es crucial la observabilidad en sistemas complejos?

El monitoreo y la observabilidad son enfoques complementarios para la salud del sistema. Por una parte, el monitoreo recolecta métricas predefinidas para verificar si los servicios funcionan según lo esperado, mientras que la observabilidad permite entender el estado interno del sistema a través de sus salidas, facilitando la investigación de problemas desconocidos o áreas dentro de dichos sistemas que requieran de una atención especial.



La observabilidad es de suma importancia en sistemas complejos, ya que las arquitecturas modernas pueden presentar fallos impredecibles que el monitoreo tradicional no puede anticipar. Entre sus principales ventajas tenemos:

- Reduce el tiempo de identificación y resolución de problemas
- Acelera la innovación entre equipos en cuanto a la estabilidad y seguridad de sus productos.
- Fomenta una cultura de responsabilidad global en cuanto a los proyectos a desarrollar.

¿Cómo puede la experiencia del desarrollador impactar el éxito de DevOps en una organización?

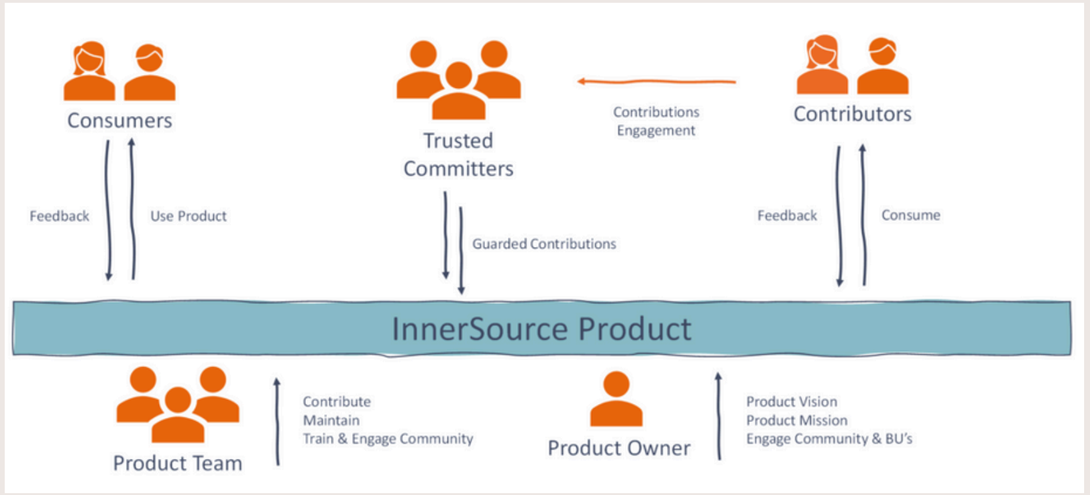
La experiencia de un desarrollador afecta significativamente al éxito de DevOps mediante la madurez técnica y una comprensión integral del ciclo de vida del software. Los desarrolladores experimentados anticipan posibles problemas en los pipelines de CI/CD, escriben código más mantenible y reducen la brecha entre desarrollo y operaciones con su conocimiento de los desafíos de implementación.

Los desarrolladores seniors también aportan una valiosa perspectiva sobre cómo el código afecta a todo el sistema y pueden fomentar la colaboración entre equipos. Su capacidad para guiar a miembros junior del equipo crea un efecto multiplicador que eleva la madurez general de DevOps.

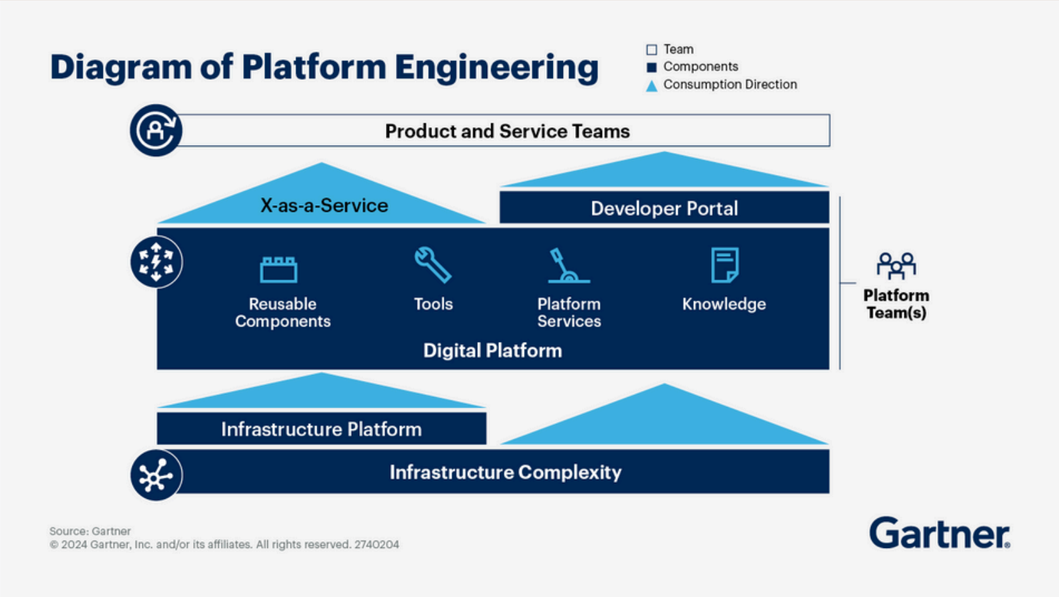
¿Cómo InnerSource puede ayudar a reducir silos dentro de una organización?

El InnerSource es un acercamiento al desarrollo de software en el que se aplican principios y prácticas de proyectos de código abierto (open source) dentro de las limitaciones de una organización o empresa. Esto consiste en tratar proyectos internos como si fuesen de código abierto, permitiendo a diferentes equipos de dicha entidad a contribuir en el desarrollo del mismo manteniendo una clara pertinencia de su respectiva área.

Este enfoque permite reducir silos mediante factores claves como la colaboración entre diferentes equipos, garantización de un desarrollo transparente con lo que múltiples integrantes puedan tener un conocimiento claro de su funcionamiento, crear componentes reutilizables e incluso llegar al desarrollo completo de librerías especializadas a la empresa, por último, fomenta prácticas saludables en cuanto al desarrollo como un uso más extenso de 'pull requests'.



¿Qué rol juega la ingeniería de plataformas en mejorar la eficiencia y la experiencia del desarrollador?



La ingeniería de plataformas es una disciplina que se enfoca en la construcción y el mantenimiento de plataformas internas de desarrollo que se encargan de proveer herramientas, infraestructura y la automatización necesarias para que los equipos de desarrollo de software operen de forma más eficiente en sus flujos de trabajo.

Esencialmente dichos profesionales crean plataformas como un producto teniendo en cuenta a desarrolladores como sus usuarios finales, sirviendo una abstracción adecuada a cuestiones empresariales técnicas.

Considerando lo explicado sobre esta especialidad, notamos que posee un papel de alta relevancia en cuanto a eficiencia y experiencia de un equipo de desarrolladores se trata; más en específico la ingeniería de plataforma se encarga en gran parte de reducir la carga cognitiva de los desarrolladores permitiendoles enfocarse unicamente en programar, elimina el trabajo repetitivo o código 'boilerplate' gracias a la automatización, permite establecer buenas prácticas como InnerSource e incrementa la construcción de un software más seguro y estandarizado.

Gracias