certiprof®

DEVOPS ADVANCED

PROFESSIONAL CERTIFICATION

DAPC™ Versión 072021

DEVOPS ADVANCED PROFESSIONAL CERTIFICATION DAPC™

DAPC™ Versión 072021

¿Quién es Certiprof®?

Certiprof® es una entidad certificadora fundada en los Estados Unidos en 2015, ubicada actualmente en Sunrise, Florida.

Nuestra filosofía se basa en la creación de conocimiento en comunidad y para ello su red colaborativa está conformada por:

- **Nuestros Lifelong Learners (LLL)** se identifican como Aprendices Continuos, lo que demuestra su compromiso inquebrantable con el aprendizaje permanente, que es de vital importancia en el mundo digital en constante cambio y expansión de hoy. Independientemente de si ganan o no el examen.
- Las universidades, centros de formación, y facilitadores en todo el mundo forman parte de nuestra red de aliados **CPLS (Certified Partner For Learning Solutions).**
- **Los autores (co-creadores)** son expertos de la industria o practicantes que, con su conocimiento, desarrollan contenidos para la creación de nuevas certificaciones que respondan a las necesidades de la industria.
- **Personal Interno:** Nuestro equipo distribuido con operaciones en India, Brasil, Colombia y Estados Unidos está a cargo de superar obstáculos, encontrar soluciones y entregar resultados excepcionales.

Nuestras Afiliaciones

Memberships

itcc
IT CERTIFICATION COUNCIL

Agile Alliance®

ANSI MEMBER

Digital badges issued by

Credly
by Pearson

IT Certification Council - ITCC

Certiprof® es un miembro activo de ITCC.

Una de las ventajas de hacer parte del ITCC es como líderes del sector colaboran entre sí en un formato abierto para explorar maneras nuevas o diferentes formas de hacer negocios que inspiran y fomentan la innovación, estableciendo y compartiendo buenas prácticas que nos permiten extender ese conocimiento a nuestra comunidad.

Certiprof ha contribuido a la elaboración de documentos blancos en el Career Path Ways Taskforce, un grupo de trabajo que se implementó internamente para ofrecer a los estudiantes la oportunidad de saber qué camino tomar después de una certificación.

Algunos de los miembros del ITCC

- **IBM**
- **CISCO**
- **ADOBE**
- **AWS**
- **SAP**
- **GOOGLE**
- **ISACA**

Agile Alliance

Certiprof® es un miembro corporativo de Agile Alliance.

Al unirnos al programa corporativo Agile Alliance, continuamos empoderando a las personas ayudándolas a alcanzar su potencial a través de la educación. Cada día, brindamos más herramientas y recursos que permiten a nuestros socios formar profesionales que buscan mejorar su desarrollo profesional y sus habilidades.

https://www.agilealliance.org/organizations/certiprof/

Credly

Esta alianza permite que las personas y empresas certificadas con Certiprof® cuenten con una distinción a nivel mundial a través de un distintivo digital.

Credly es el emisor de insignias más importante del mundo y empresas líderes en tecnología como IBM, Microsoft, PMI,Nokia, la Universidad de Stanford, entre otras, emiten sus insignias con Credly.

Empresas que emiten insignias de validación de conocimiento con Credly:

- **IBM**
- **Microsoft**
- **PMI**
- **Universidad de Stanford**
- **Certiprof**

Insignias Digitales

Insignias Digitales:
¿Qué Son?

Según el estudio del IT Certification Council (ITCC), años atrás, la gente sabía muy poco sobre las insignias digitales. Hoy, grandes empresas e instituciones educativas de todo el mundo expiden insignias.

Las insignias digitales contienen metadatos detallados sobre quién las ha obtenido, las competencias requeridas y la organización que las ha expedido. Algunas insignias incluso están vinculadas a las actividades necesarias para obtenerlas.

Para las empresas e instituciones educativas, las insignias y la información que proporcionan son tan importantes que muchas decisiones, como las de contratación o admisión, se basan en los datos que aportan.

¿Por qué son importantes?

- **Facilidad de Compartir y Verificar Logros:**

  Las insignias digitales permiten a los profesionales mostrar y verificar sus logros de manera instantánea y global. Según un informe de Credly, **los perfiles de LinkedIn con insignias digitales reciben un 40% más de atención por parte de reclutadores y empleadores.**

- **Visibilidad en Plataformas Digitales:**

  En una encuesta realizada por Pearson y Credly, el **85%** de los usuarios que obtuvieron insignias digitales **las compartieron en LinkedIn,** y el **75%** reportó que esto mejoró su **credibilidad profesional en sus redes.** Además, el **76%** de los empleadores encuestados afirmó que las insignias digitales les ayudan a identificar rápidamente habilidades específicas.

¿Por qué son importantes?

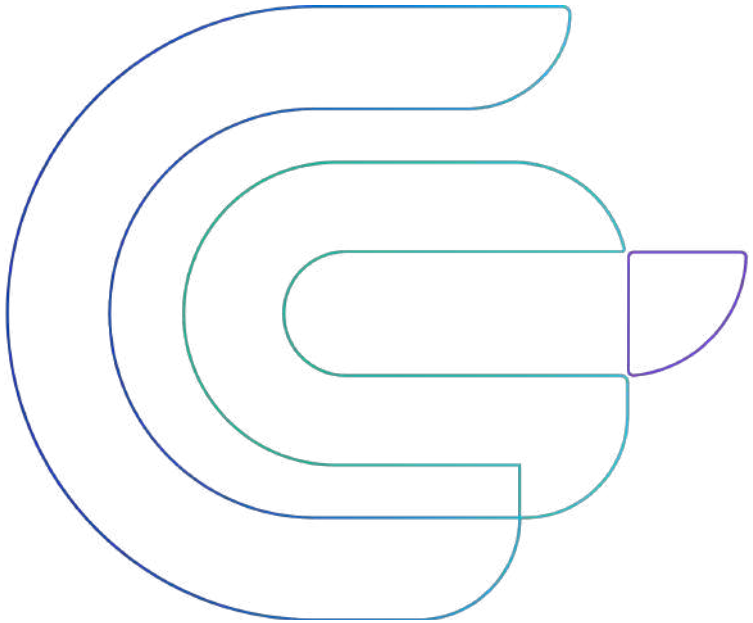
- **Impacto en la Contratación:**

  Un estudio de la **Asociación Internacional de Gestión de Proyectos (PMI)** encontró que los candidatos que muestran insignias digitales de gestión de proyectos tienen **un 60%** más de probabilidades de ser contratados en comparación con aquellos que solo mencionan sus habilidades sin verificación digital.

¿Por qué son importantes?

- **Empoderamiento de la Marca Personal:**

La visibilidad y verificación instantánea proporcionada por las insignias digitales permiten a los profesionales no solo demostrar sus habilidades, sino también construir una marca personal fuerte. Según un estudio de LinkedIn, los profesionales que utilizan insignias digitales tienen un 24% más de probabilidades de avanzar en sus carreras.
La certificación y las insignias digitales no son solo una validación del conocimiento, sino también una herramienta poderosa para la mejora continua y la empleabilidad. En un mundo donde el aprendizaje permanente se ha convertido en la norma, estas credenciales son clave para el desarrollo profesional y la competitividad en el mercado laboral global.



- No todas las insignias son iguales, y en **Certiprof,** estamos comprometidos con ofrecerte más que un simple reconocimiento digital. Al obtener una insignia emitida por certiprof, estarás recibiendo una validación de tu conocimiento respaldada por una de las entidades líderes en certificación profesional a nivel mundial.

- **Da el siguiente paso y obtén la insignia que te abrirá puertas y te posicionará como un experto en tu campo.**

¿Por qué es importante obtener su certificado?

- **Prueba de experiencia:** Su certificado es un reconocimiento formal de las habilidades y conocimientos que ha adquirido. Sirve como prueba verificable de sus cualificaciones y demuestra su compromiso con la excelencia en su campo.

- **Credibilidad y reconocimiento:** En el competitivo mercado laboral actual, las empresas y los compañeros valoran las credenciales que le distinguen de los demás. Un certificado de una institución reconocida, como Certiprof, proporciona credibilidad instantánea e impulsa su reputación profesional.

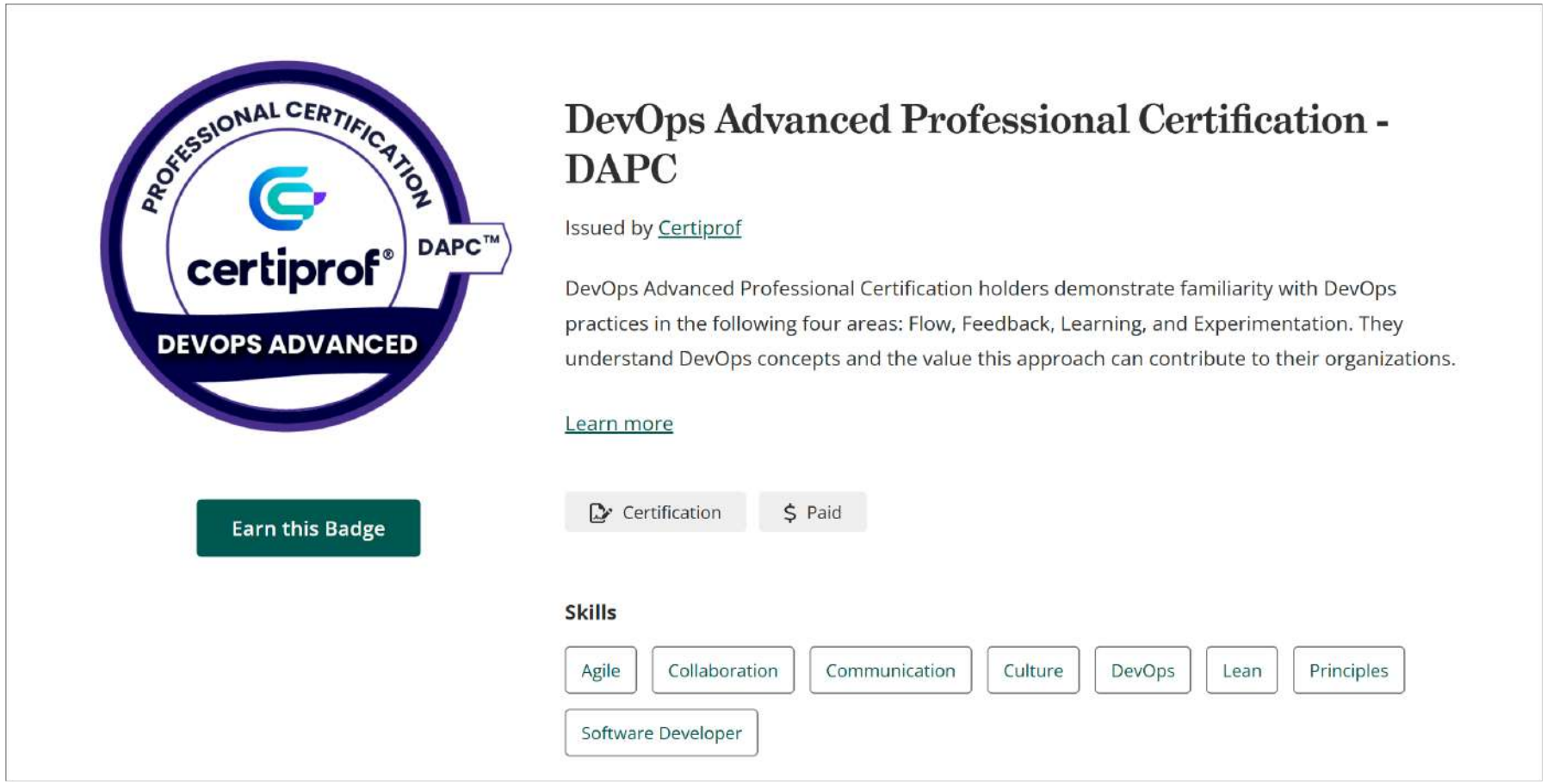
- **Avance profesional:** Tener tu certificado puede abrirte las puertas a nuevas oportunidades. Ya se trate de un ascenso, un aumento de sueldo o un nuevo puesto de trabajo, las certificaciones son un factor diferenciador clave que los empleadores tienen en cuenta a la hora de evaluar a los candidatos.

¿Por qué es importante obtener su certificado?

- **Oportunidades de establecer contactos:** Poseer un certificado le conecta con una red de profesionales certificados. Muchas organizaciones cuentan con grupos de antiguos alumnos o de trabajo en red en los que puede compartir experiencias, intercambiar ideas y ampliar su círculo profesional.

- **Logro personal:** Obtener una certificación es un logro importante, y su certificado es un recordatorio tangible del trabajo duro, la dedicación y el progreso que ha realizado. Es algo de lo que puede sentirse orgulloso y mostrar a los demás.

Insignia



https://www.credly.com/org/certiprof/badge/devops-advanced-professional-certification-dapc.1

Aprendizaje Permanente

- Certiprof ha creado una insignia especial para reconocer a los aprendices constantes.
- Para el 2024, se han emitido más de 1,000,000 de estas insignias en más de 11 idiomas.

Propósito y Filosofía

- Esta insignia está destinada a personas que creen firmemente en que la educación puede cambiar vidas y transformar el mundo.
- La filosofía detrás de la insignia es promover el compromiso con el aprendizaje continuo a lo largo de la vida.

Acceso y Obtención de la Insignia

- La insignia de Lifelong Learning se entrega sin costo a aquellos que se identifican con este enfoque de aprendizaje.
- Cualquier persona que se considere un aprendiz constante puede reclamar su insignia visitando:

https://certiprof.com/pages/certiprof-lifelong-learning

COMPARTE Y VERIFICA TUS LOGROS DE APRENDIZAJE FÁCILMENTE

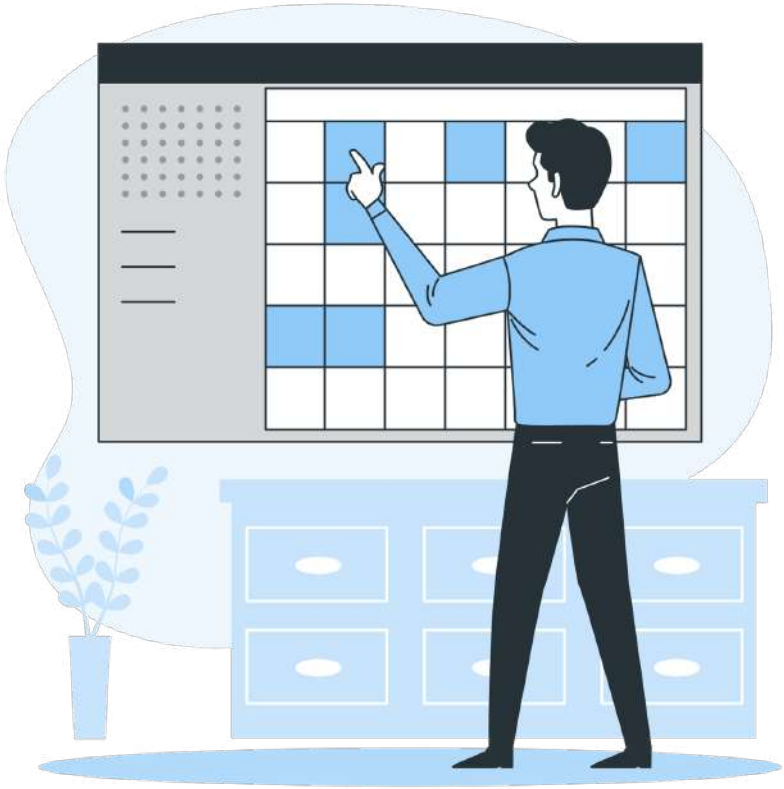
#DAPC #certiprof

certiprof®

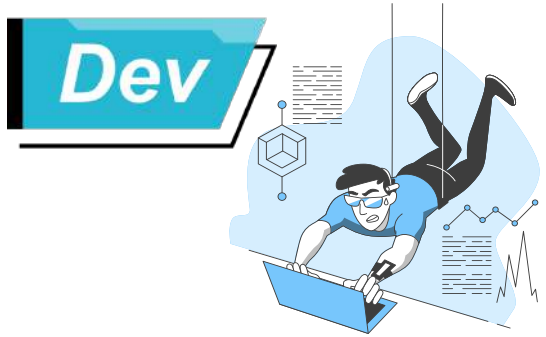
Parte 1: Introducción

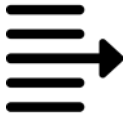
Agenda

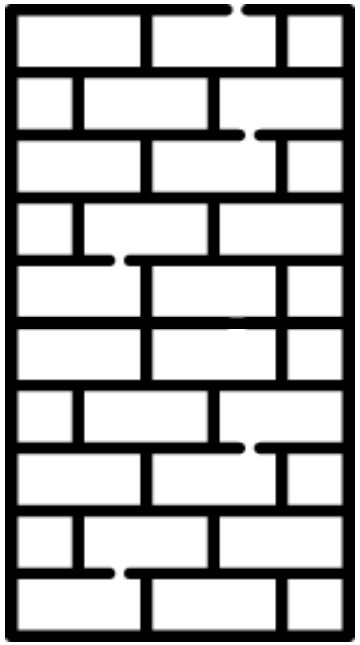
1. Surgimiento
2. Antecedentes
3. Conceptos Base de Devops
   - Definiciones base
   - Pilares y Principios de DevOps
   - Organización - Team Topologies
   - Integrando Ops en Dev



...Nació el Muro de la Confusión







Objetivos:
Entrega de nuevas funcionalidades (idealmente de calidad)

Cultura de Producto (Software Development)

Quiere entregar nuevas funcionalidades

Objetivos:
Garantizar la "ejecución" de las aplicaciones (estabilidad)

Cultura de Servicio (Deployment, Monitoring, Support)

Quiere mantener la estabilidad y los servicios disponibles

Introducción

2007: Patrick Debois

2008: Andrew Shafer

2009: John Allspaw

2009: DevOpsDays, Gante, Bélgica. Patrick Debois

Introducción

DevOps es un movimiento originado en 2007, cuando Patrick Debois, un consultor freelance, estuvo analizando el sector IT. Su objetivo principal era ganar experiencia en todas las perspectivas en la cadena de valor IT. Tiene un blog bastante interesante.

En 2007 mientras Debois estaba en un proyecto de consultoría para el gobierno de Bélgica para la migración de un data center se frustra debido a los conflictos que se producían entre desarrolladores y administradores de sistemas. Debido a esto él decide proponer una solución.

En Agosto del 2008, en Agile2008 conference, Andrew Clay Shafer y Patrick Debois se conocen. Shafer habló sobre la "infraestructura Agile" y sólo una persona atiende: Patrick Debois. Shafer, al ver que no había más que un asistente, piensa que no había interés en este tema salta esta sesión y más tarde, Debois tiene una larga conversación con él sobre el tema en el pasillo.

Introducción

De esta y otras conversaciones surgieron todas las típicas frustraciones y conflictos que existen entre operaciones y desarrollo, así como sus motivos. Ambos crearon un grupo en Google denominado "Agile System Administrators" para continuar el debate.

En 2009 John Allspaw y Paul Hammond presentan "10 deploy per Day"en la O'Relly Velocity 09 Conference. La principal premisa era enfocarse en asegurar que Dev y Ops trabajaban juntos y de forma crossfunctional a través de las herramientas y procesos ágiles.

Esto inspiró a Debois creando el evento "DevOpsDays" en Bélgica (DOD). Decide hacer exactamente lo que indica la charla y toma de aquí el concepto DevOps (Dev and Ops Cooperation at Flickr). La conferencia de puertas abiertas tiene lugar en Octubre de este año y se continua la discusión vía Twitter. El movimiento empieza a surgir.

Introducción

Al mismo tiempo, la integración continua está en su momento de auge dentro del espectro Agile. Y había mucho movimiento en cuanto al continuous deployment sobre todo con la aparición del libro "Continuous delivery".

Mientras en paralelo la industria IT empiezan a tener fuerza otras metodologías como Operation Management, Lean e IT service management.

Este movimiento formado por debates, conferencias, twitter y lentamente tomó atención de la industria. IBM, CA Technologies, HP y BMC empiezan a aplicarlo.

En una entrevista de InfoQ en Abril del 2012, Debois admite que el nombre no fue intencional, que simplemente el título original era muy largo y escribió "DevOpsDay" para acortar. De aquí emergió el nombre DevOps.

Fuente: https://danielvillahermosa.wordpress.com/2019/11/27/la-evolucion-de-devops/

1.2 Antecedentes

Varios movimientos importantes en gestión y tecnología convergen para preparar el escenario para el movimiento DevOps.

La aplicación de los principios del Lean al flujo de valor de la tecnología deriva en "Tres Formas":

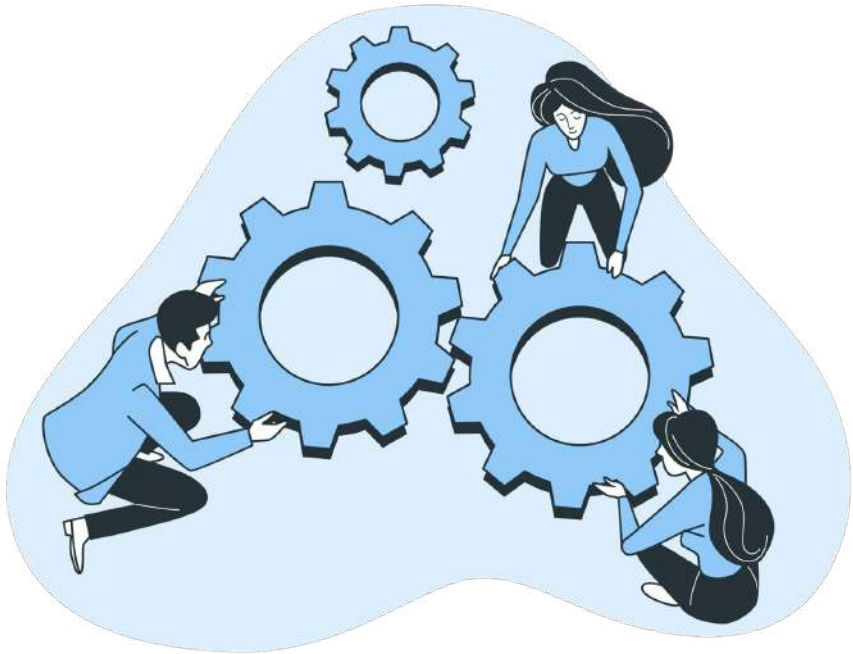
1. Flujo
2. Feedback
3. Aprendizaje y Experimentación Continuos

1.2 Antecedentes

DevOps es el resultado de la aplicación de los principios más confiables, desde el dominio de la manufactura , el liderazgo industrial hasta el flujo de valor de TI.

DevOps cuenta con varios conceptos provenientes de diferentes fuentes de conocimiento:

- ✓ Lean Manufacturing
- ✓ Teoría de las Restricciones
- ✓ Sistema de Producción de Toyota
- ✓ Ingeniería de resiliencia
- ✓ Movimiento de la Entrega Continua
- ✓ Organizaciones que aprenden
- ✓ Cultura de seguridad
- ✓ Factores humanos y muchos otros



1.3 Conceptos Base de Devops

Conceptos Base de Devops

- Entrega continua
- Infraestructura Ágil
- Lead time
- Kata
- Wip
- Deuda técnica

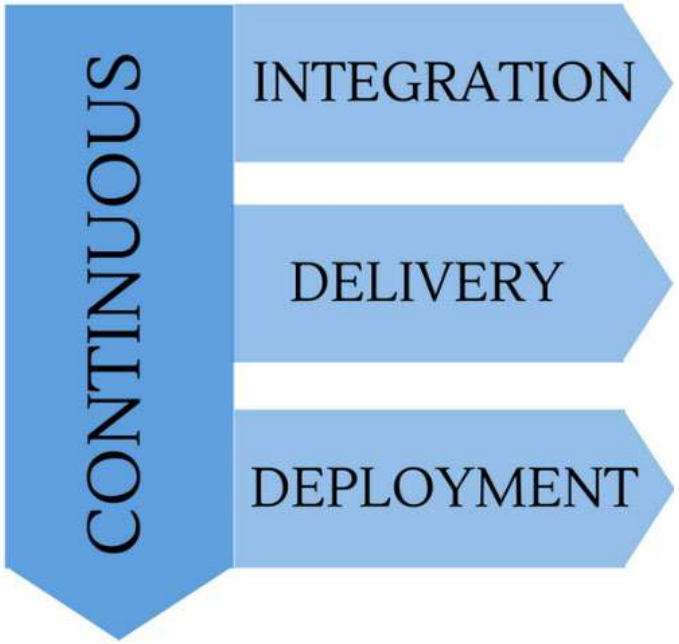
1.3 Conceptos Base de Devops

La **Entrega Continua** es una extensión de la integración continua para garantizar que se pueda liberar nuevos cambios para sus clientes rápidamente de forma sostenible.

¿Dónde está la diferencia entre Integración y Entrega Continua?

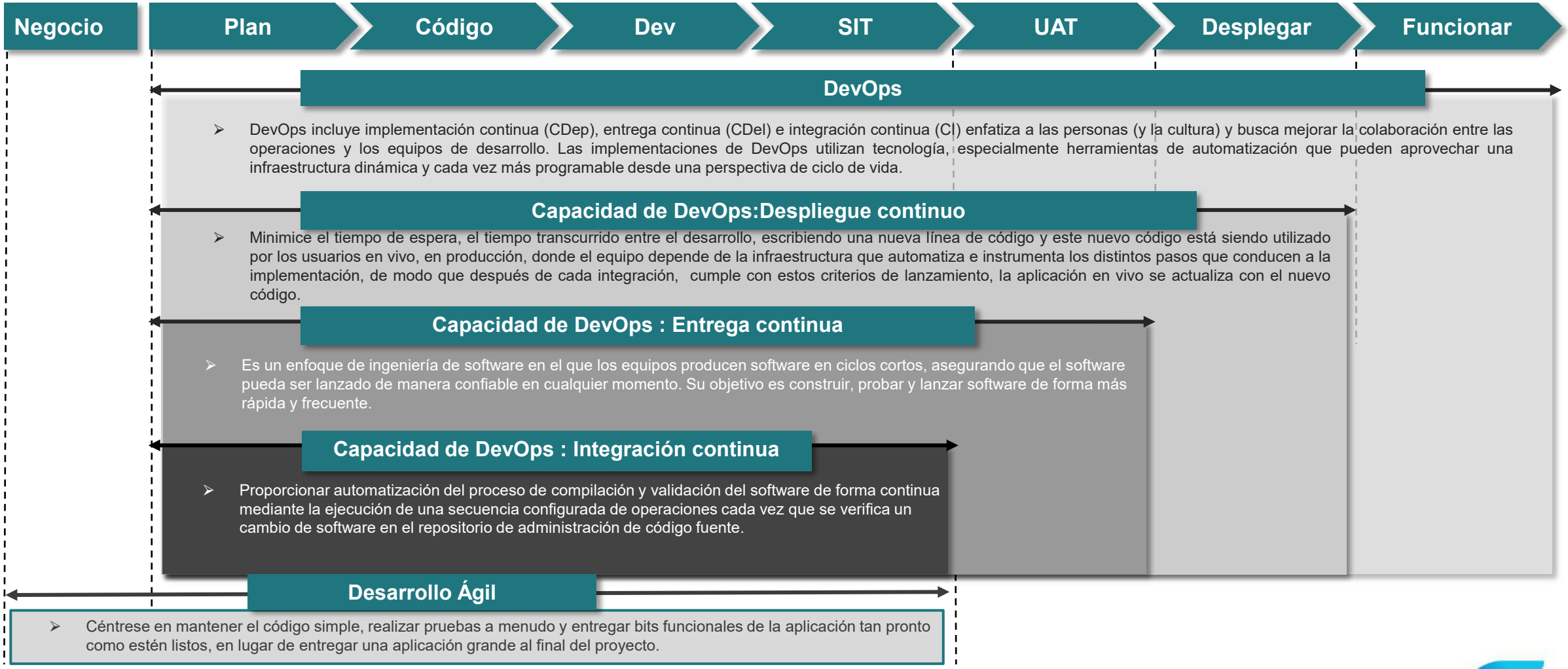
La **entrega continua** define la función de una "pipeline de despliegue" para garantizar que el código y la infraestructura estén siempre en un estado implantable y que todo el código registrado en la rama principal o master pueda ser implantado con seguridad en producción. La integración continua forma parte de la entrega continua y del despliegue continuo.

El **despliegue continuo** es una extensión de la entrega continua, excepto por el hecho de que las liberaciones ocurren automáticamente en Producción.



DevOps complementa a Agile para aumentar la eficiencia del modelo de entrega desde la gestión de relaciones comerciales hasta el producto final en producción.

DevOps

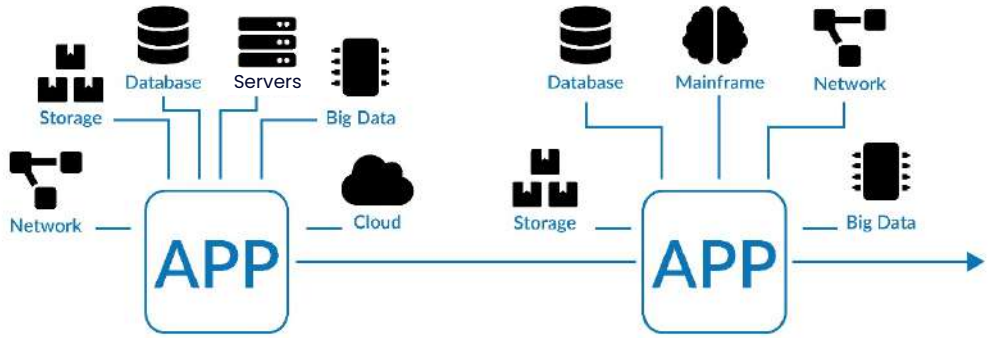


1.3.1 Conceptos Base: Infraestructura Ágil

La infraestructura como código es el enfoque para definir la infraestructura de computación y red, que se utiliza de técnicas de administración de código fuente, siendo tratado como cualquier sistema de software.

Este código puede ser mantenido en el sistema de control de versiones para permitir auditabilidad y construcción reproducible, sujeto a las prácticas de prueba y a la disciplina total de Entrega Continua. La infraestructura como código se basa en algunas prácticas:

- Utilizar archivos de definición
- Sistemas y procesos auto documentados
- Versión de todos los elementos
- Probar continuamente sistemas y procesos
- Pequeños cambios en lugar de grandes lotes
- Mantener los servicios disponibles continuamente



"Habilita la reconstrucción del negocio a partir de la nada, además de un repositorio de código fuente, copia de seguridad de datos de aplicaciones y recursos físicos crudos" - Adam Jacob, CTO de Chef.

1.3.1 Conceptos Base: Infraestructura Ágil

X como código

Infraestructura como código

Network como código

Políticas como código

Configuración como código

Seguridad como código

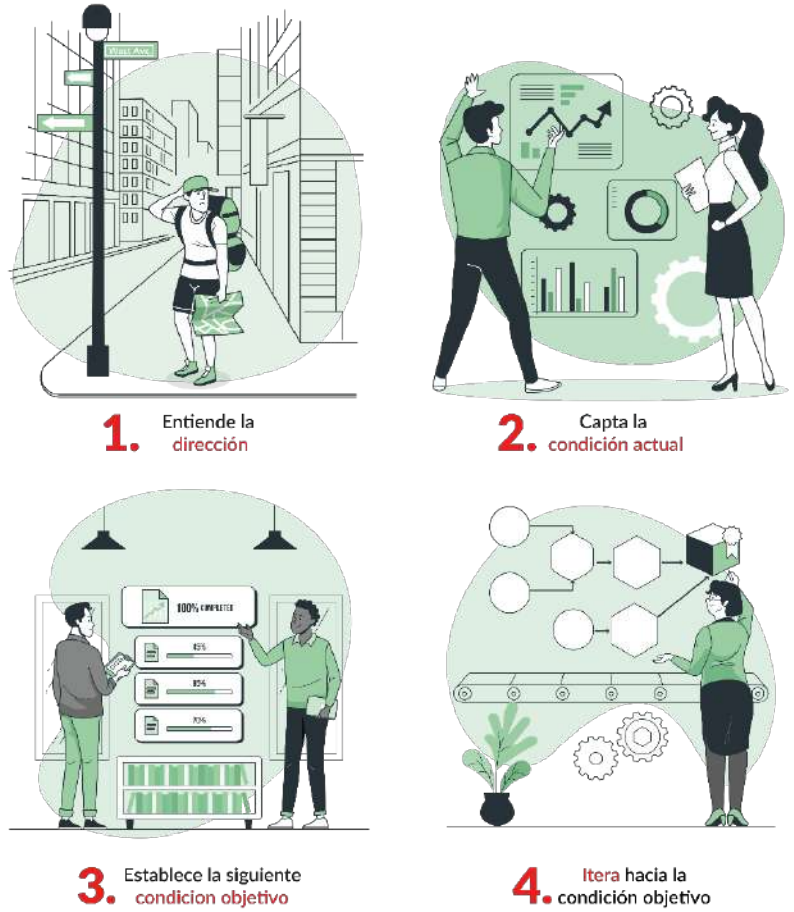
1.3.1 Conceptos Base: Kata

Los equipos generalmente no son capaces o no están dispuestos a mejorar los procesos en que operan. El resultado no es sólo que continúan sufriendo con sus problemas actuales, su sufrimiento también empeora con el tiempo.

En el Toyota Kata que, en ausencia de mejoras, los procesos no permanecen iguales, debido al caos y a la entropía, los procesos realmente se degradan con el tiempo.

Descomponga cada secuencia de trabajo en etapas específicas, automatícelas y tráigalas de forma estandarizada, que puede usarse "reflexivamente", cuando sea necesario.

Se refiere a la forma o estándar que se puede practicar para desarrollar habilidades personales y la mentalidad.



Kata son rutinas de enseñanza usadas para preservar y pasar el know-how.

1.3.1 Conceptos Base: Work in Progress (WIP)

WIP: Trabajo que ya está en el proceso de desarrollo, pero todavía no está terminado y disponible para un cliente o un usuario.

Se refiere a todos los activos o elementos de un producto o servicio que actualmente se están trabajando o esperando para ser trabajados.

Podemos limitar la multitarea cuando usamos un Tablero Kanban para gestionar nuestro trabajo, por ejemplo, codificando e imponiendo límites de WIP (work in progress) para cada columna o centro de trabajo. Podemos colocar un límite superior en el número de tarjetas que pueden estar en una columna.

Trabajo en proceso o progreso, que incluye el conjunto de grandes cantidades de elementos inacabados para los productos en un proceso de producción.

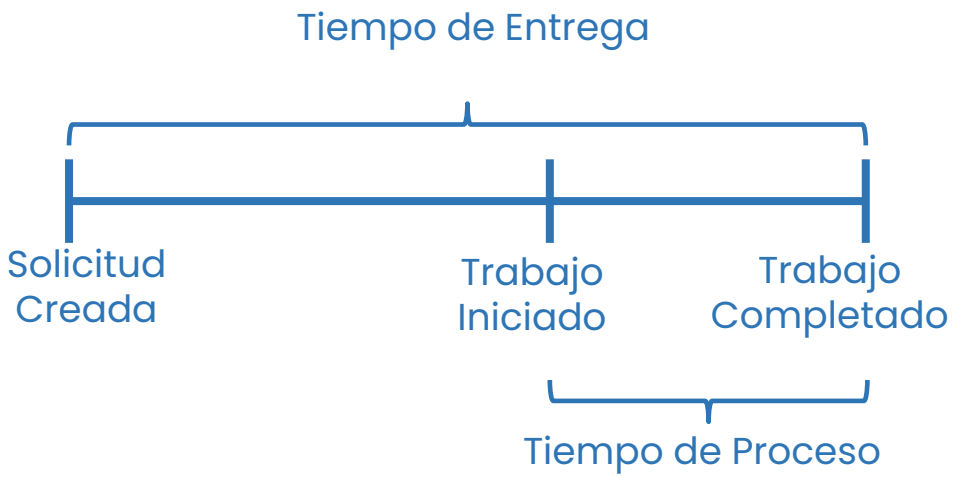
1.3.1 Conceptos Base: Lead Time

El Tiempo de Entrega (lead time) es una de las medidas comúnmente usadas para medir el rendimiento en los flujos de valor (Lean).

La proporción de tiempo de proceso vs el tiempo de entrega, sirve como una medida importante de eficiencia

La medición de tiempo de entrega comienza cuando la solicitud es creada y termina cuando es entregada al cliente.

La medición del tiempo de proceso sólo comienza cuando empezamos a trabajar en la solicitud del cliente - específicamente, omite el tiempo que el trabajo está en la cola, esperando ser procesado.



Tiempo de Entrega vs. tiempo de proceso de una operación de implementación
Fuente: Manual DevOps, capítulo 1

1.3.1 Conceptos Base: Deuda Técnica

El costo implícito del retrabajo adicional causado por la elección de una solución fácil ahora en vez de usar un enfoque mejor que llevaría más tiempo.

Fuente: Vincent Déniel on Twitter: https://www.pinterest.es/pin/798192733957711009/?d=t&mt=login

1.3.1 Conceptos Base: Deuda Técnica

La deuda técnica describe cómo las decisiones que tomamos conducen a problemas que se vuelven cada vez más difíciles de arreglar con el tiempo, reduciendo continuamente nuestras opciones disponibles en el futuro, incluso cuando se toman con prudencia, todavía generamos intereses sobre esa deuda.

- Definición inicial insuficiente
- Presiones de negocios
- Falta de proceso o comprensión
- Componentes fuertemente acoplados
- La falta de un conjunto de pruebas
- Falta de documentación
- Falta de colaboración
- El desarrollo paralelo
- Refactorización retrasada
- Falta de alineación con los estándares
- Falta de conocimiento
- Falta de propiedad
- Liderazgo tecnológico deficiente
- Cambios de especificación a la última hora

El costo implícito del retrabajo adicional causado por la elección de una solución fácil ahora en vez de usar un enfoque mejor que llevaría más tiempo.

1.3.1 Conceptos Base: Bimodal SoR y SoC

Sistemas de Registros

Los sistemas corporativos que giran nuestros negocios, donde la corrección de las transacciones y de los datos son primordiales

- Cambios con ritmo más lento
- Hacer correctamente

Sistemas de Contratación

Son sistemas orientados al cliente o utilizados por los empleados, sistemas de comercio y aplicaciones de productividad.

- Ritmo de cambios más rápidos
- Hacer rápido

Gartner recientemente popularizó la noción de TI bimodal, refiriéndose al amplio espectro de servicios que las empresas típicas soportan.

1.3.1 Conceptos Base: Bimodal SoR y SoC

SoR

Los sistemas de registro típicamente tienen un ritmo de cambio más lento y muchas veces tienen requisitos de conformidad y reglamentación (por ejemplo, SOX).

Gartner reúne estos tipos de sistemas Tipo 1, donde la organización se concentra en "hacer correctamente".

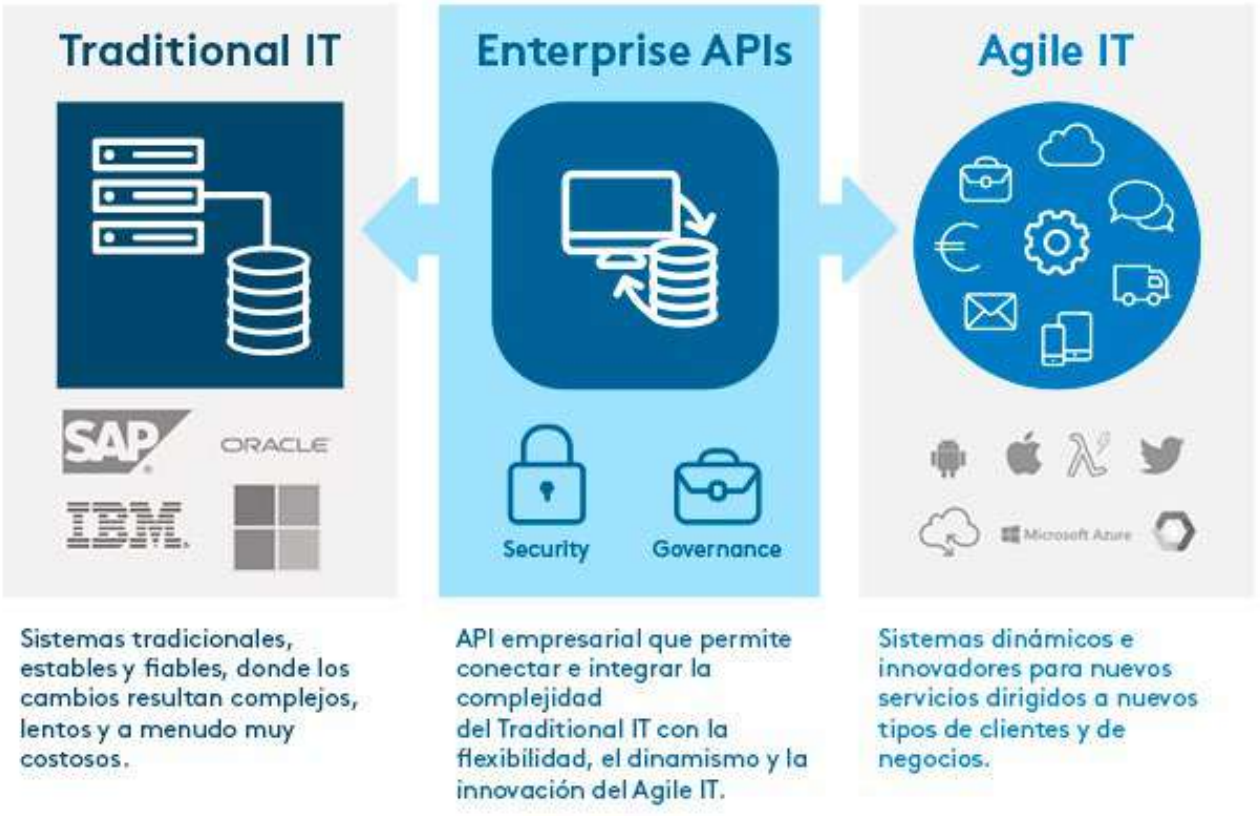
SoC

El sistema de contratación generalmente tiene un ritmo de cambio mucho mayor para soportar ciclos de feedback rápidos que le permiten realizar experimentos para descubrir cómo satisfacer mejor las necesidades del cliente.

Gartner llama a estos tipos de sistemas Tipo 2, "donde la organización se concentra en hacer rápidamente".

1.3.1 Conceptos Base: Bimodal SoR y SoC

Para complementar ambos entornos tecnológicos y alcanzar **la eficiencia del Bimodal IT** en el que conviven muchas empresas, apostamos por arquitecturas que permitan la integración de datos y la integración de aplicaciones de forma adecuada, y recomienda soluciones basadas en **Enterprise APIs** que permiten conectar e integrar la complejidad del Traditional IT con la flexibilidad, el dinamismo y la innovación del Agile IT.

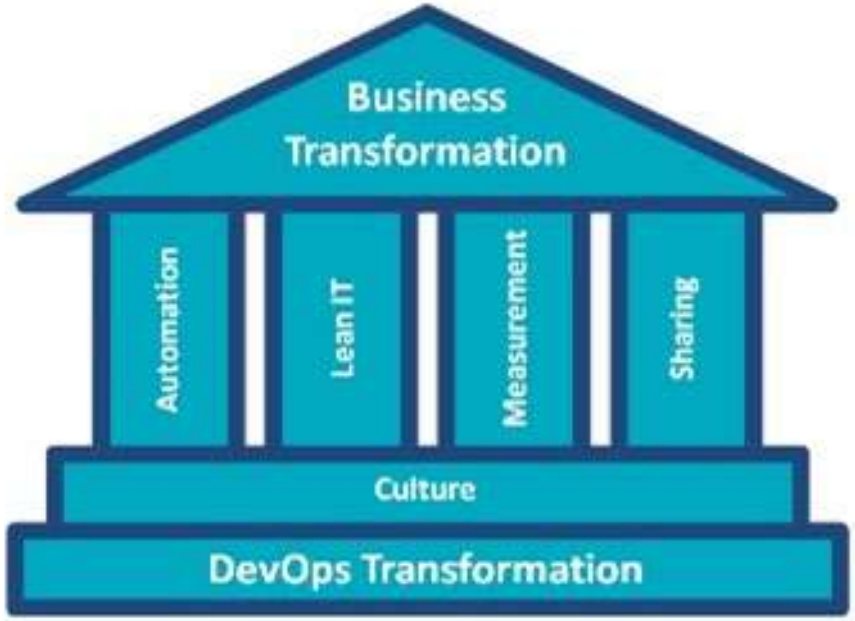


Fuente: *https://www.ithinkupc.com/es/blog/como-conseguir-la-eficiencia-en-un-entorno-bimodal-it*

Discusión en Grupos – Manifiesto Ágil

1.3.2 Pilares de DevOps

Mantenga la CALMa(S)



- **Culture**
- **Automation**
- **Lean**
- **Measurement**
- **Sharing**

1.3.2 Pilares de DevOps

Cultura

1. Sin un cambio cultural, la transformación no va a funcionar
2. Más que una metodología, es una colección de principios para romper lo silos

Automatización

1. Automatización ayuda pero no es una bala de plata
2. Puede habilitar la mejora en la calidad y la entrega continua a través de la integración continua

Lean

1. Ayuda a reducir los desperdicios en el proceso
2. Ayuda a poner foco en el objetivo final que quieras lograr desde el punto de vista del negocio

Métricas

1. Debes Medir todo - repetidamente
2. Debes adaptarte y planificar áreas de mejor basado en tu lectura de esas métricas

Sharing (Colaboración)

1. Uno de los principales aspectos de DevOps
2. Ayuda a romper los silos
3. Un habilitador fantástico para ampliar los ciclos de retroalimentación

Discusión en Grupos – Manifiesto Ágil

1.3.3 Organización

Arquetipos Organizativos

Hay tres tipos principales de Estructuras Organizacionales que informan sobre cómo proyectamos nuestros flujos de valor DevOps con la Ley Conway en mente:

Orientación Funcional	Orientación Matricial	Orientación del Mercado
• Las organizaciones orientadas a funciones optimizan la experiencia, la división del trabajo o reduciendo el costo.	• Las organizaciones orientadas en matrices combinan orientación funcional y de mercado.	• Las organizaciones orientadas al mercado optimizan para responder rápidamente a las necesidades del cliente.

1.3.3 Organización

Regla del “2 pizzas Team”

- Garantiza que el equipo tenga una comprensión clara y compartida del sistema en el que están trabajando
- Limita la tasa de crecimiento del producto o servicio en el que trabajó
- Descentraliza el poder y permite la autonomía
- Liderar un 2PT es una manera para que los empleados ganen alguna experiencia de liderazgo en un entorno donde el fracaso no tiene consecuencias catastróficas

1.3.3 Organización

Ley de Conway

Melvin Conway observó que la forma en que las organizaciones estaban estructuradas tenía un fuerte impacto en los sistemas que creaban.

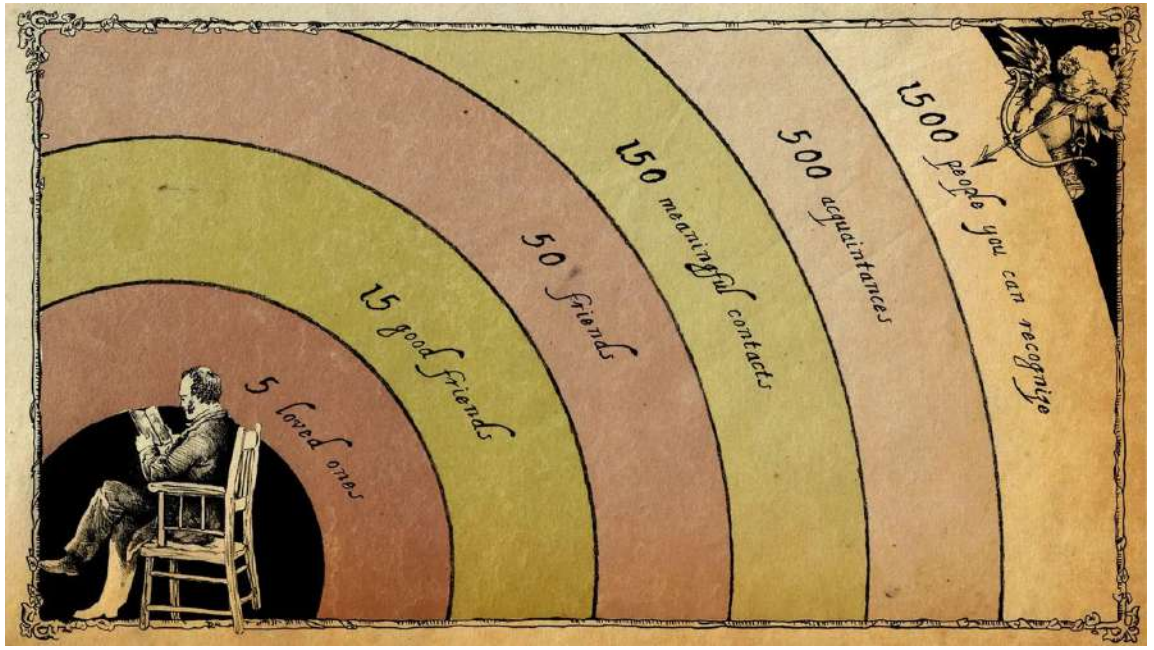
Estas observaciones llevaron a lo que ahora se conoce como Ley de Conway, que afirma que "las organizaciones que proyectan sistemas... son constreñidas para producir dibujos que son copias de las estructuras de comunicación de esas organizaciones... Cuanto mayor es la organización, menor es la flexibilidad y más pronunciado el fenómeno".

"Cualquier empresa que proyecta un sistema (definición más amplia aquí que sólo sistemas de información), inevitablemente produce un proyecto cuya estructura es una copia de la estructura de comunicación de la organización".

1.3.3 Organización

El número de Dunbar

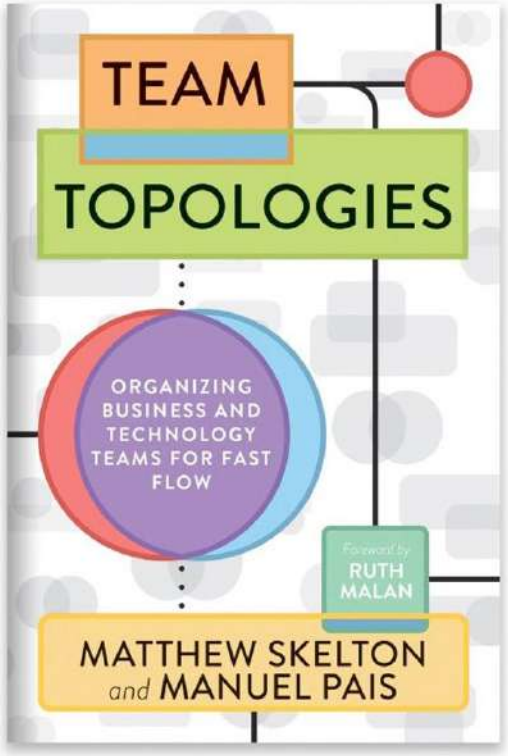
"Según el antropólogo británico Robin Dunbar, su teoría de que los humanos solo pueden mantener 150 amistades ha resistido 30 años de escrutinio. Dunbar se convenció de que había una relación entre el tamaño del cerebro y el tamaño de los grupos a través de sus estudios de primates no humanos. Esta proporción se trazó utilizando neuroimágenes y observación del tiempo dedicado al aseo, un comportamiento social importante de los primates. Dunbar concluyó que el tamaño, en relación con el cuerpo, del neocórtex, la parte del cerebro asociada con la cognición y el lenguaje, está relacionado con el tamaño de un grupo social cohesionado. Esta relación limita la complejidad que puede manejar un sistema social".



El círculo más íntimo son solo cinco seres queridos, llegando a un máximo de 1500 personas que puedes reconocer (Crédito: Emmanuel Lafont).

Team Topologies – Organizing business and technology teams for fast flow

Autores:
Matthew Skelton
Manuel Pais

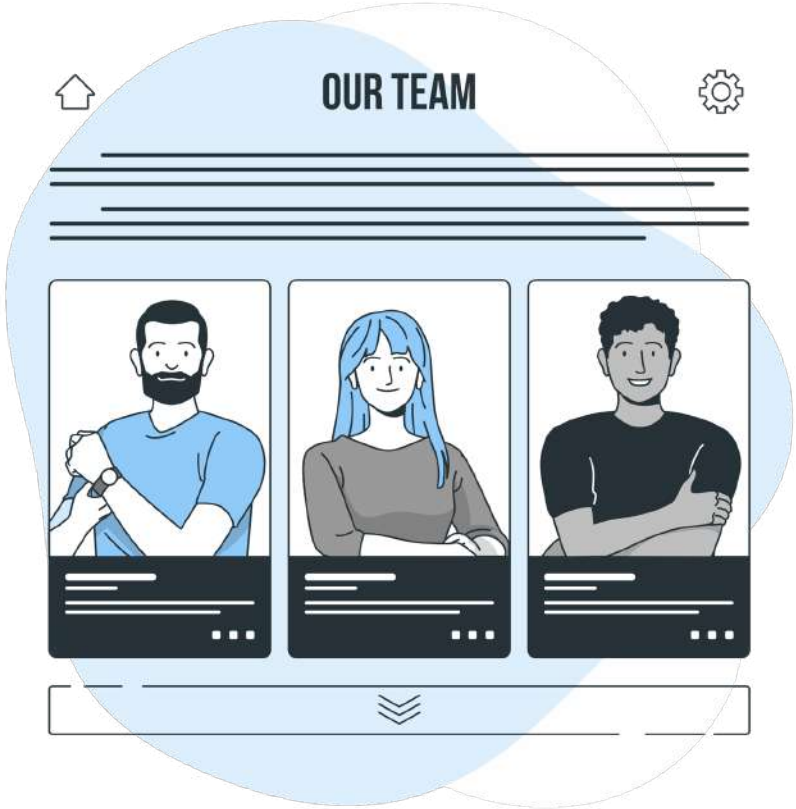


1.3.3 Organización

Cuatro topologías fundamentales

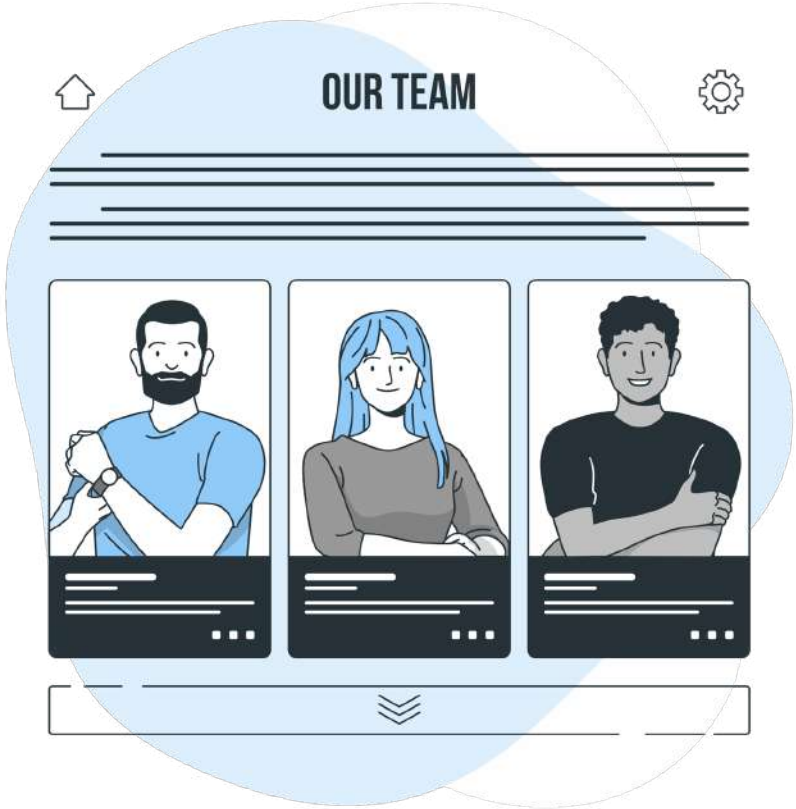
A continuación detallaremos las cuatro topologías fundamentales del equipo, incluido el comportamiento y las capacidades esperados. Estas son:

- Equipo alineado al flujo del producto o servicio
- Equipo de plataforma
- Equipo habilitador
- Equipo de subsistema complicado



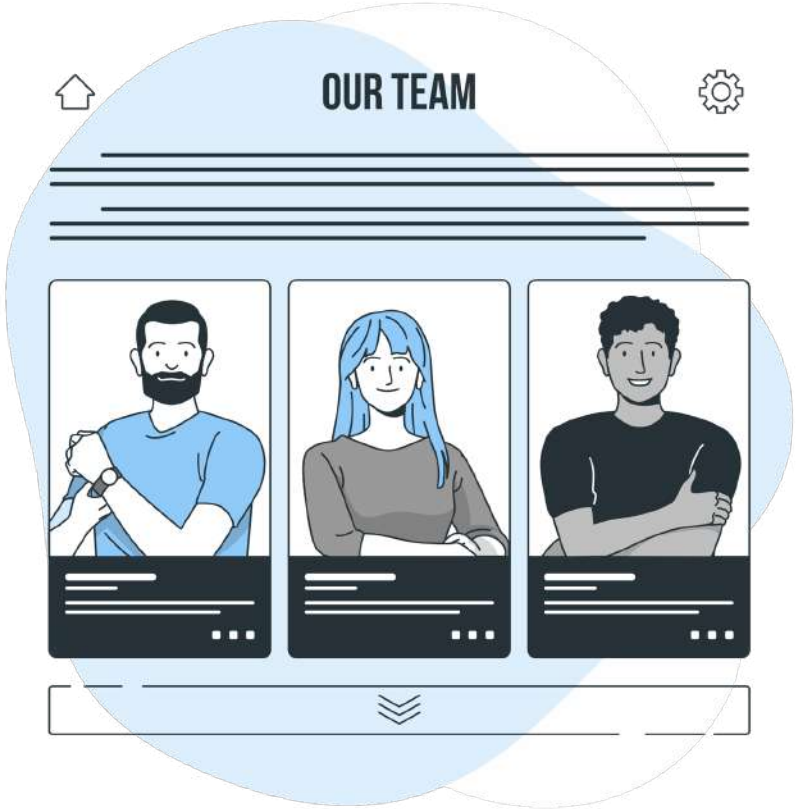
1.3.3 Organización

- **Equipo alineado al flujo del producto o servicio:** un equipo alineado con el flujo principal del producto o servicio, con una combinación de habilidades multifuncionales y la capacidad de ofrecer incrementos significativos sin esperar a otro equipo (algunos llamarían a estos equipos "equipos de productos o funciones" pero hablar de flujos tiene más sentido)
- **Equipo de la plataforma:** un equipo que trabaja en la plataforma subyacente y da soporte a los equipos alineados con el flujo en la entrega. La plataforma simplifica la tecnología que de otro modo sería compleja y reduce la carga cognitiva para los equipos que la utilizan (una buena plataforma es "lo suficientemente grande")



1.3.3 Organización

- **Equipo habilitador:** un equipo que ayuda a otros equipos a adoptar y modificar software como parte de un período de transición o aprendizaje
- **Equipo de subsistemas complicados:** un equipo con un mandato especial para un subsistema que es demasiado complicado para ser tratado por un equipo normal alineado con el flujo o un equipo de plataforma. Opcional y solo se usa cuando es realmente necesario

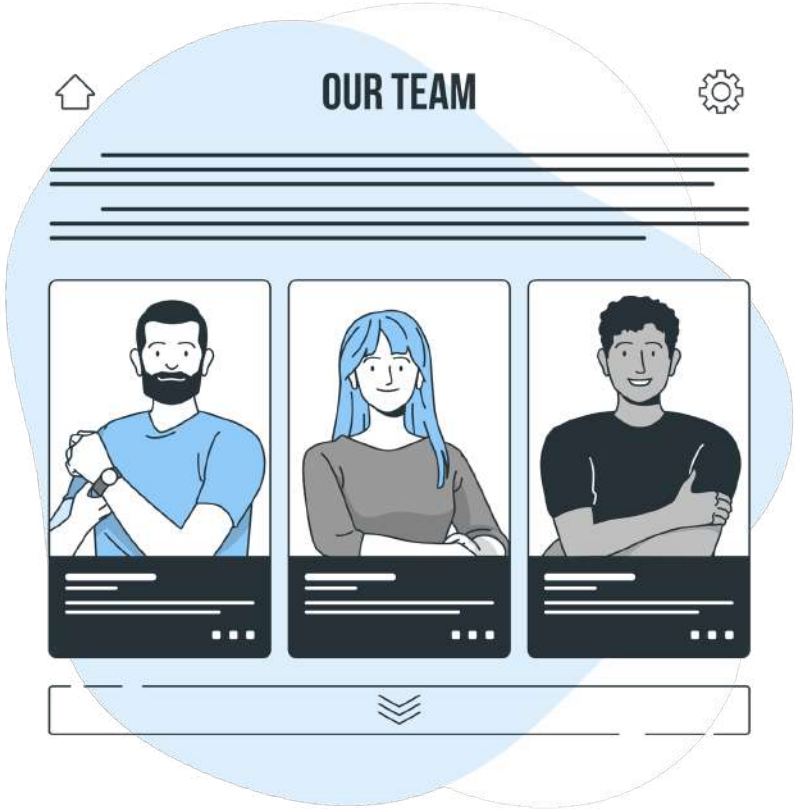


1.3.3 Organización

Tres modos de interacción

Modos de interacción en equipo. Los modos de interacción principales para las 4 topologías de equipo fundamentales son:

- Colaboración: trabajar en estrecha colaboración con otro equipo
- X-as-a Service: consumir o proporcionar algo con una colaboración mínima
- Facilitar: ayudar (o ser ayudado por) otro equipo para eliminar los impedimentos



Tipos de equipos fundamentales

Equipo alineado al flujo del producto o servicio

Equipo habilitador

Equipo del subsistema complicado

Equipo de plataforma

Modos de interacción en equipo

Colaboración

Xaas

Facilitar

1.3.3 Organización: Equipos Auto Gestionados

Forma I, Forma T y Forma E

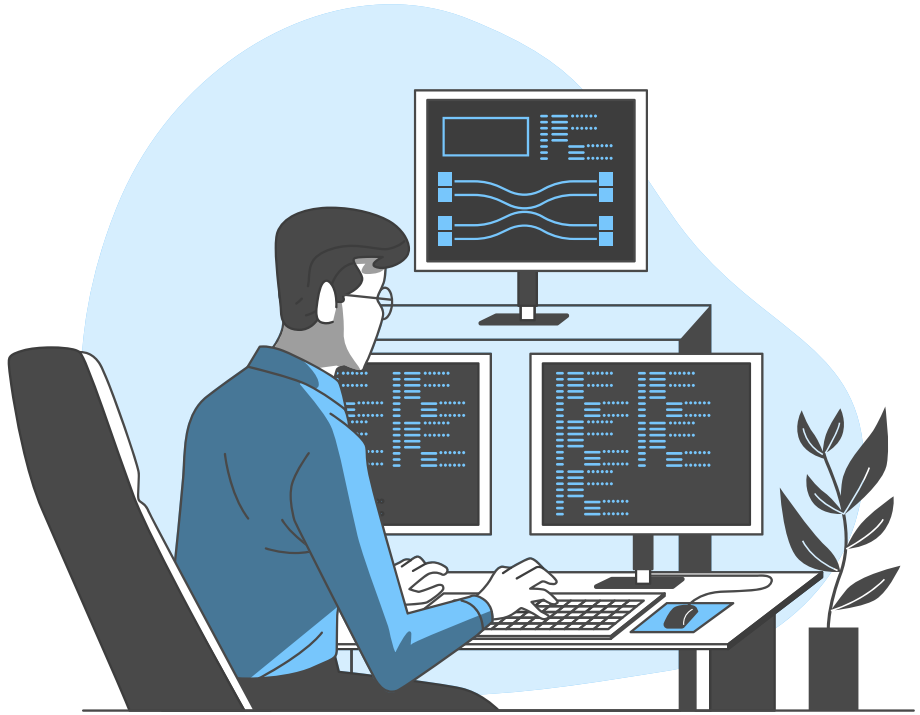
Cuando los departamentos se especializan demasiado, causan trabajos orientados en silos. Cualquier actividad operativa compleja, a continuación, requiere múltiples transferencias y fugas entre las diferentes áreas de la infraestructura, llevando a plazos más largos.

A través de la capacitación cruzada y de las crecientes habilidades de ingeniería, los generalistas pueden tener una mayor magnitud de órdenes de trabajo que sus contrapartes especializadas, y también mejora el flujo general de trabajo, removiendo colas y tiempo de espera.

1.3.3 Organización: Equipos Auto Gestionados

Forma I: Expertos

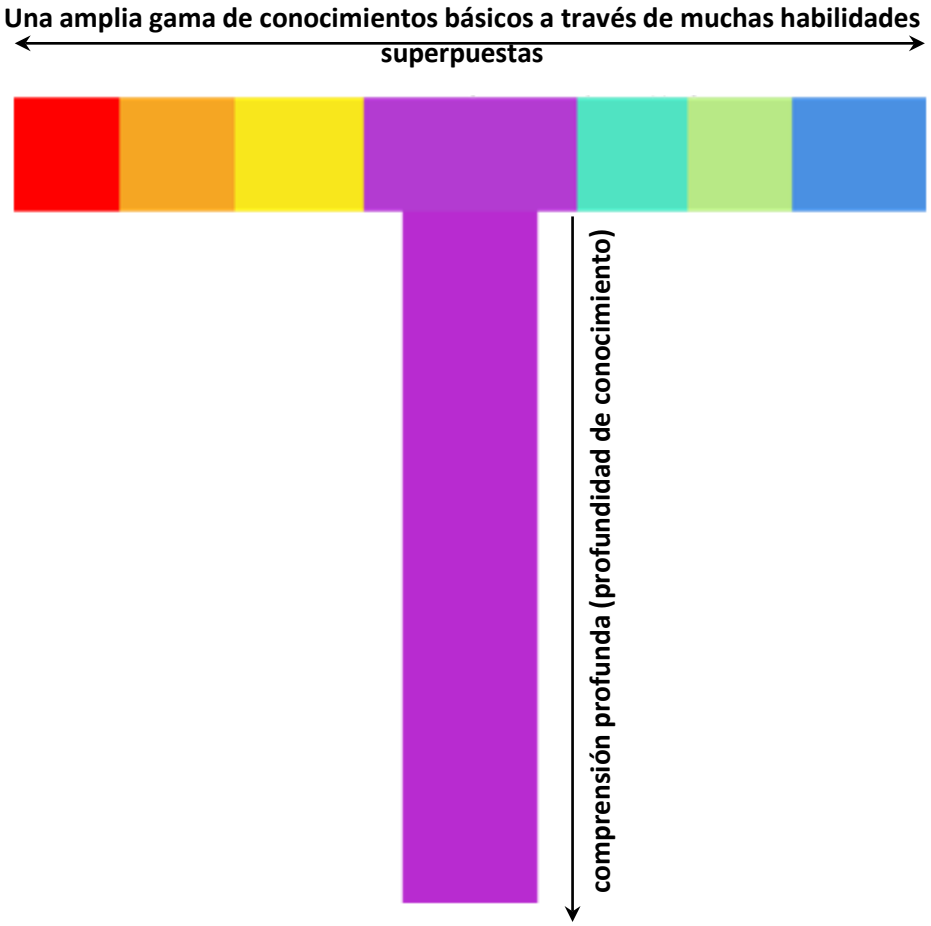
- Especialización profunda en un área
- Pocas habilidades o experiencia en otras áreas
- Crea cuellos de botellas rápidamente
- Insensible a los residuos y al impacto en el flujo
- Evita la planificación de flexibilidad o absorción de variabilidad



1.3.3 Organización: Equipos Auto Gestionados

Forma T: Generalistas

- Especialización profunda en un área
- Excelentes habilidades en muchas áreas
- Puede intensificar para quitar los cuellos de botella
- Sensible a los residuos y al impacto del flujo
- Ayuda a tornar la planificación flexible y absorbe la variabilidad

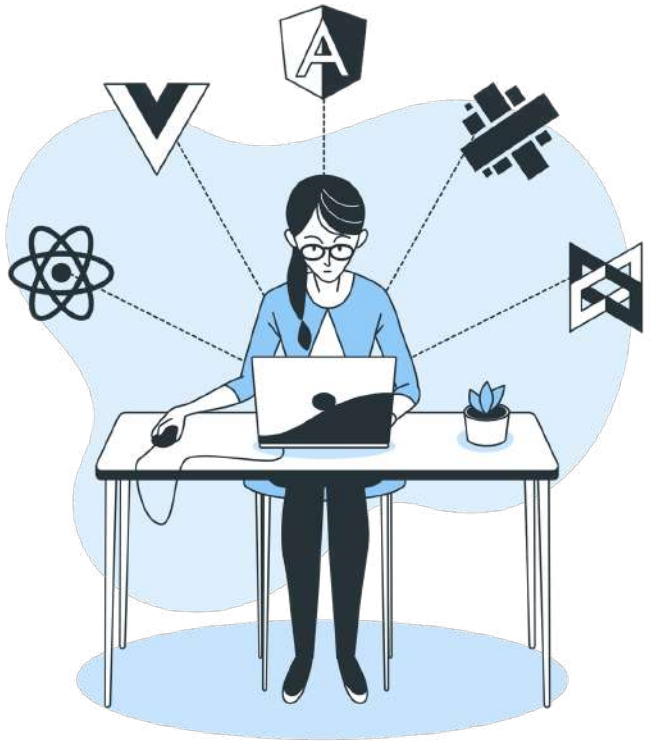


1.3.3 Organización: Equipos Auto Gestionados

Forma E

- Especialización profunda en pocas áreas
- Experiencia cruzada en varias áreas
- Presentan habilidades de ejecución
- Siempre está innovando
- Potencial casi sin límites

Cuanto mayor sea el conocimiento general y las habilidades especiales más necesarias dentro de una persona, mejor.



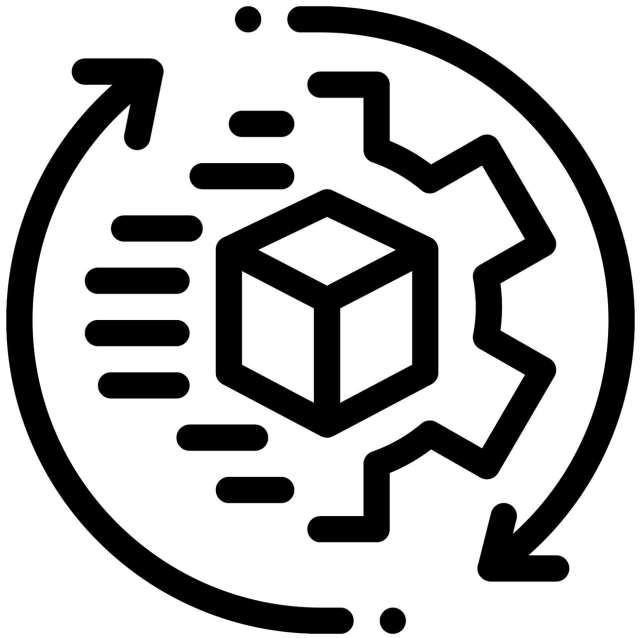
1.3.4 Integrando Ops en Dev

Integrando Ops en Dev

El objetivo es tener resultados orientados al mercado, donde muchos pequeños equipos pueden proporcionar un valor para el cliente de manera rápida e inmediata.

Podemos crear resultados más orientados al mercado, integrando mejor las capacidades Ops en equipos Dev, haciéndolas más eficientes y productivas.

Ops puede mejorar significativamente la productividad de los equipos de Dev en toda la organización, además de permitir una mayor colaboración y resultados organizativos.



1.3.4 Integrando Ops en Dev

Podemos usar tres estrategias:

- Creando habilidades de autoservicio que permita su activación por los desarrolladores
- Incorporando a los ingenieros de Ops en los equipos de servicio
- Desarrollando vínculos entre Ops y los equipos de servicio

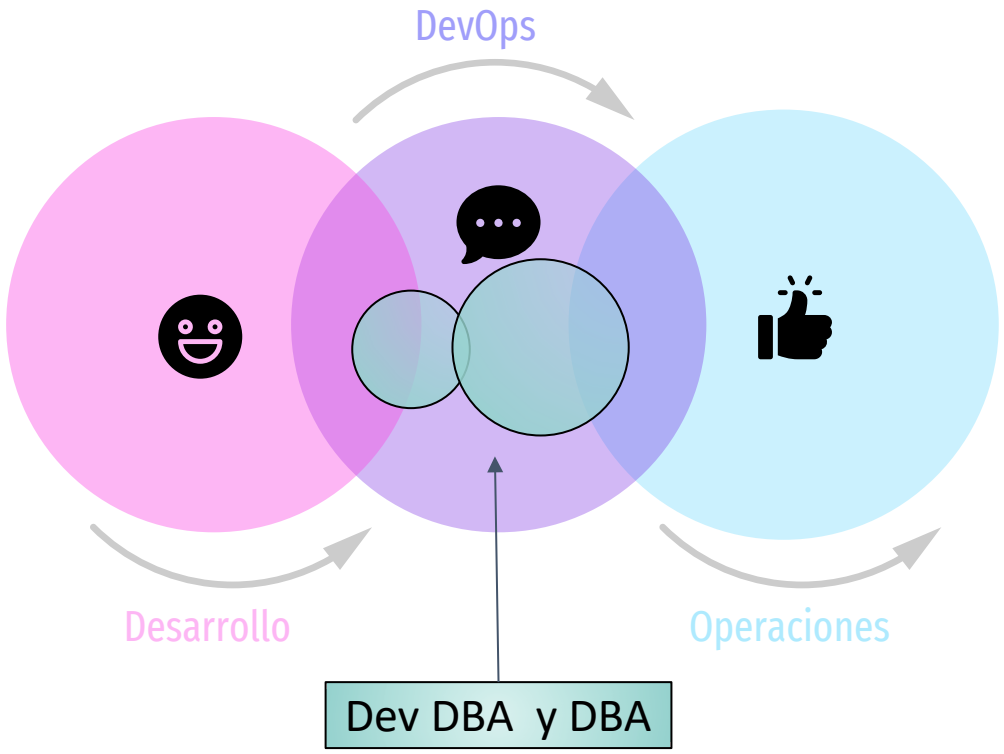
1.3.4 Integrando Ops en Dev

Cree capacidades de autoservicio para uso de los desarrolladores

Habilitamos a los equipos de Dev a pasar más tiempo creando funcionalidades para sus clientes... Al hacer esto, habilitamos a los equipos de productos a obtener lo que necesitan, cuando lo necesitan, además de reducir la necesidad de comunicación y coordinación.

Una manera de proporcionar resultados orientados al mercado es que las Operaciones creen un conjunto de plataformas centralizadas y herramientas de servicios, las que cualquier equipo de Dev pueda usar para volverse más productivo:

- Obtener entornos de producción
- Pipelines de despliegue
- Herramientas de prueba automatizadas
- Paneles de producción de telemetría
- Entre otros



https://web.devopstopologies.com/

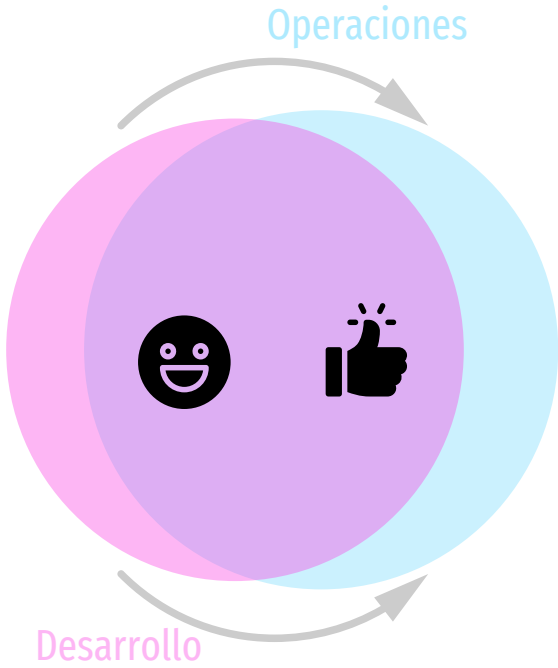
1.3.4 Integrando Ops en Dev

Incorpore los ingenieros de Ops en los equipos de servicio

Habilitando equipos de productos para llegar a ser más autosuficientes incorporando ingenieros de operaciones dentro de ellos, reduciendo así su dependencia de operaciones centralizadas.

Las prioridades de los ingenieros de operaciones se dirigen casi enteramente a los objetivos de los equipos de productos, cuando se integran en los equipos del Dev.

Como resultado, los ingenieros de Ops están más conectados a sus clientes internos y externos.

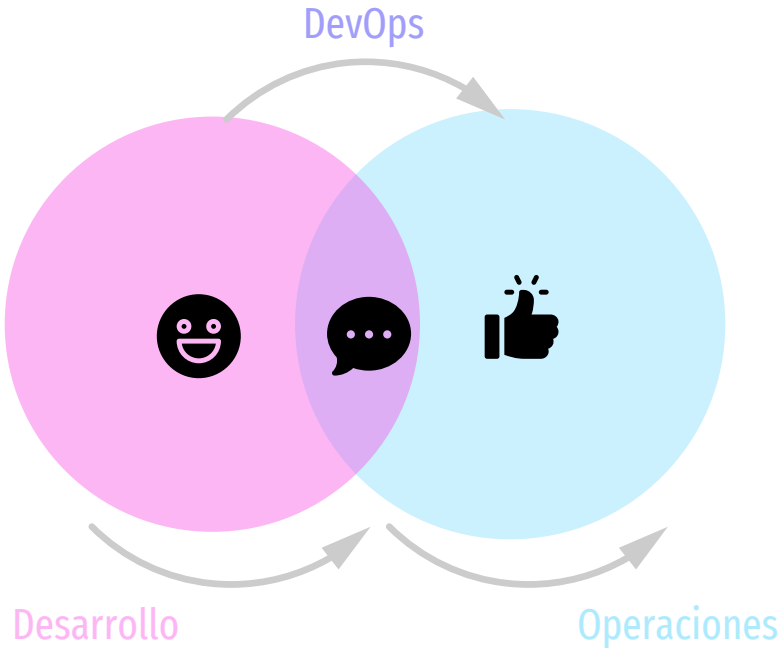


https://web.devopstopologies.com/

1.3.4 Integrando Ops en Dev

Desarrolle conexiones entre Ops y otros equipos cuando la incorporación de Ops no sea posible

Por una variedad de razones, como costo y falta de recursos, tal vez no podamos integrar ingenieros OPS en cada equipo de productos.



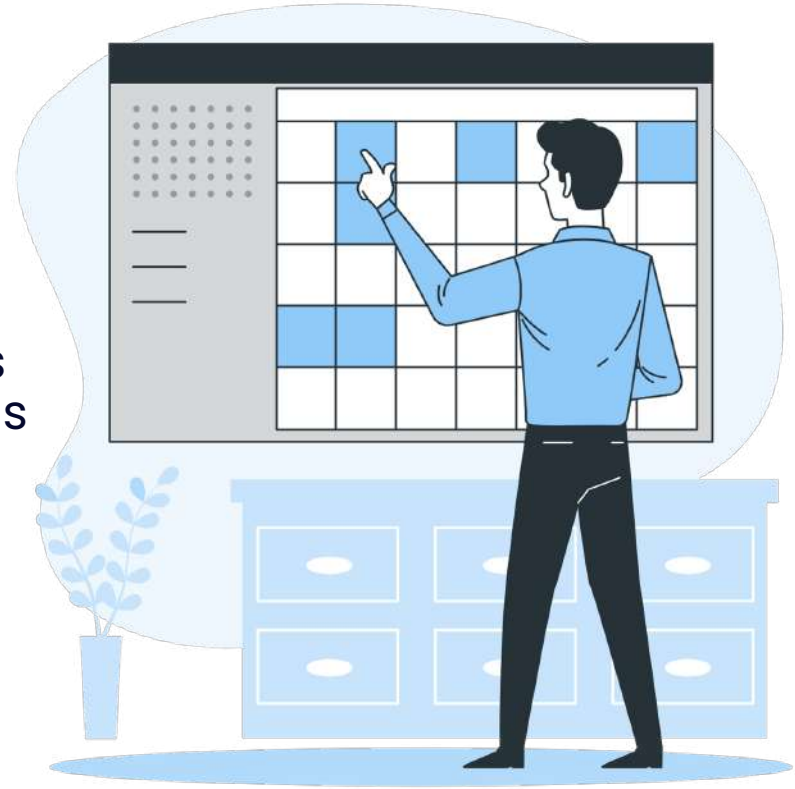
https://web.devopstopologies.com/

Discusión en Equipos – DevOps Topologies

Parte 2: Principios de DevOps – 3 Ways of DevOps

Agenda

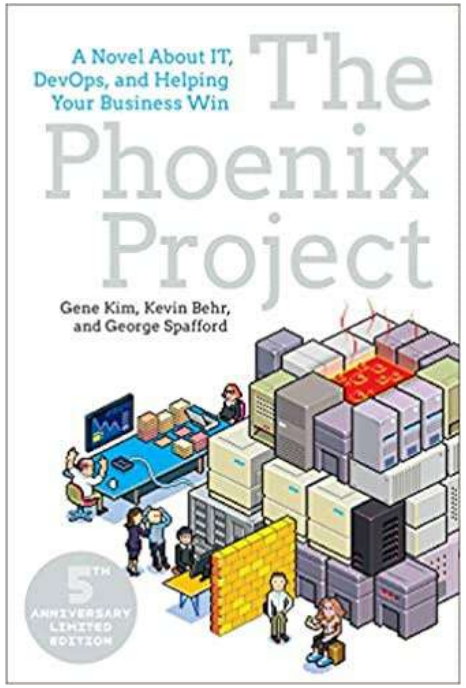
1. Las “3 ways of DevOps”
   - La primera vía y las competencias necesarias
   - La segunda vía y las competencias necesarias
   - La tercera vía y las competencias necesarias



Bibliografía

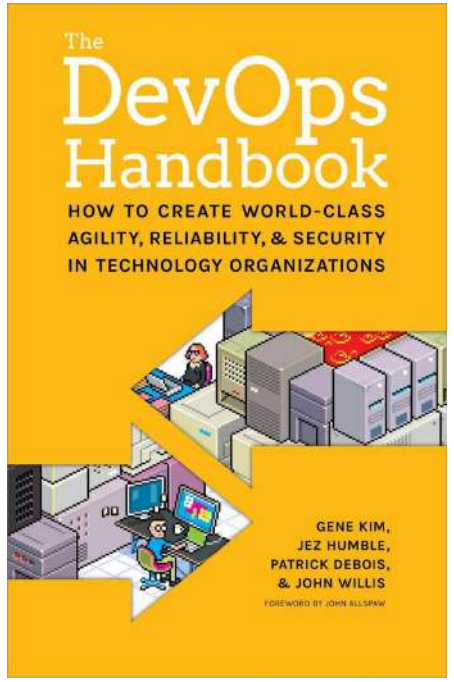
The Phoenix Project (A Novel About IT, DevOps, and Helping Your Business Win)

Autores:
Gene Kim
Kevin Behr
George Spafford



The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations

Autores:
Gene Kim
Jez Humble
Patrick Debois
John Wills



Los Autores

Gene Kim es un CTO, investigador y autor galardonado en múltiples ocasiones de *The Phoenix Project: A Novel About IT, DevOps y Helping Your Business Win* y T*he Visible Ops Handbook*. Es fundador de *IT Revolution* y es anfitrión de las conferencias *DevOps Enteprise Summit.*

Patrick Debois es un consultor de TI independiente que está reduciendo la brecha entre proyectos y operaciones mediante el uso de técnicas ágiles, en desarrollo, gestión de proyectos y administración de sistemas.

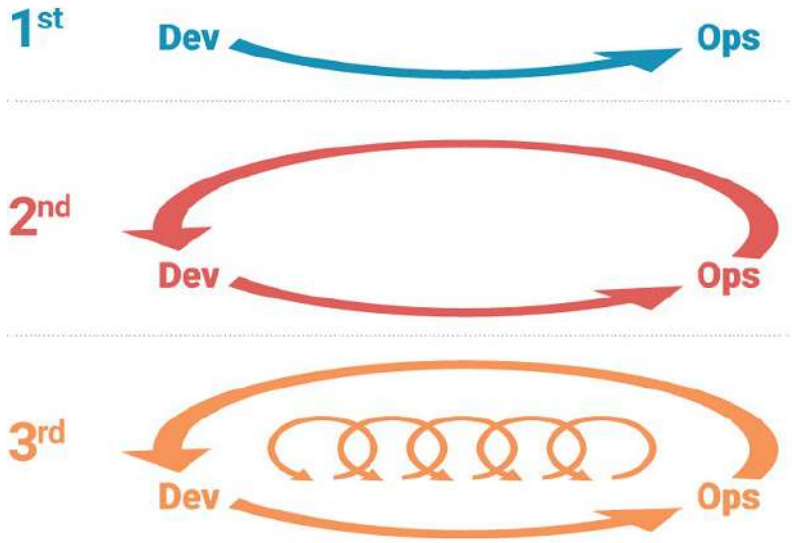
Jez Humble es coautor de *Continuous Delivery*, ganador del premio Jolt, y del innovador *Lean Enterprise*. Su enfoque es ayudar a las organizaciones a entregar software valioso y de alta calidad de manera frecuente y confiable mediante la implementación de prácticas de ingeniería efectivas.

John Willis ha trabajado en la industria de la gestión de TI durante más de treinta y cinco años. Es autor de seis IBM Redbooks y fue el fundador y arquitecto jefe de *Chain Bridge Systems*. Actualmente es evangelista en **Docker, Inc**.

Principios de DevOps

“DevOps no se trata de una tecnología, DevOps se trata de un problema de negocios” - Patrick Debois.

La mejor manera de entender DevOps es destilarlo en 3 principios popularizados por el Proyecto Phoenix y explicados en detalle en el Devops Handbook: Pensamiento sistémico, retroalimentación y experimentación y aprendizaje continuo.

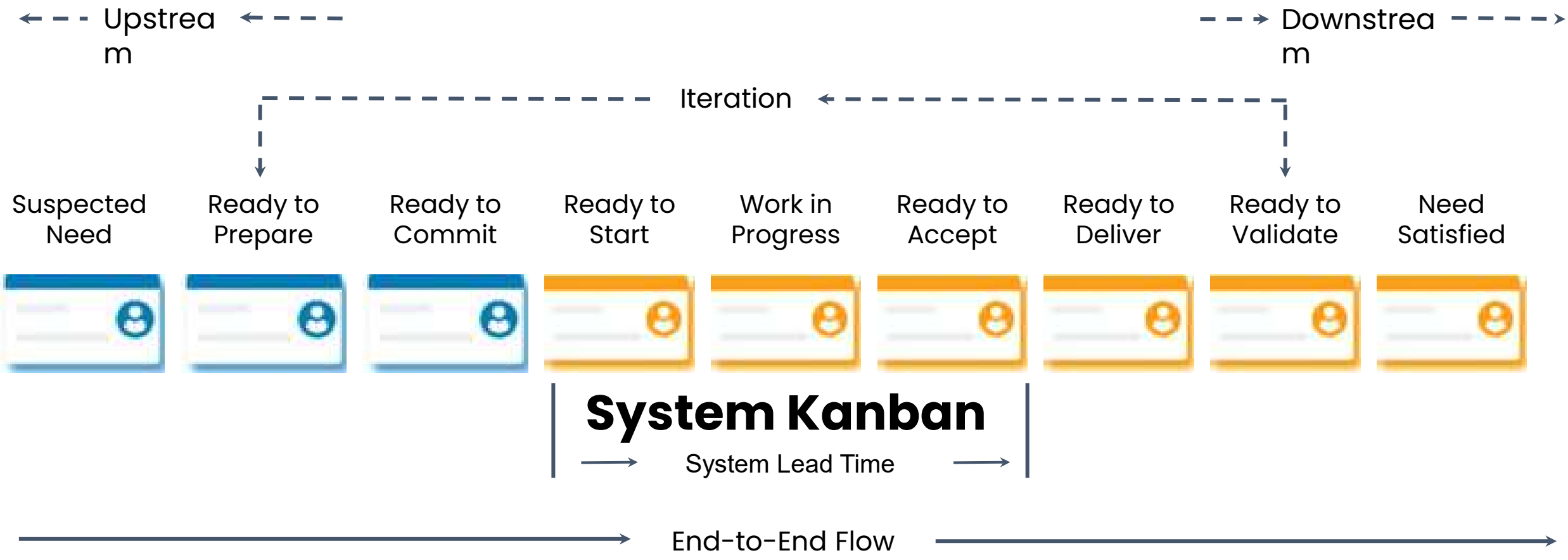


La Primera Forma

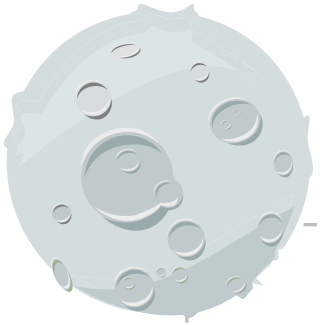
The First Way: Los Principios de Flujo

"La primera forma enfatiza el rendimiento de todo el sistema, a diferencia del rendimiento de un silo específico de trabajo o departamento. DevOps trasciende los departamentos y muestra el valor general para el cliente" - Gene Kim.

3.4 Integración Continua



Competencias Necesarias - Capabilities

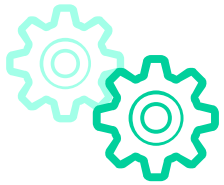


01 Identificar las cadenas de valor

02 Hacer visible nuestro trabajo

03 Limitar el trabajo en procesos (WIP)

04 Reducir el tamaño de los lotes de trabajo



05 Reducir el número de transferencias (HANDOFFS)

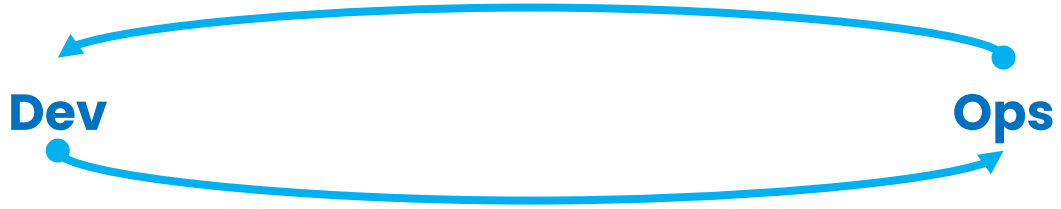
06 Identificar continuamente y elevar nuestras restricciones

07 Eliminar obstáculos y desperdicios en la cadena de valor

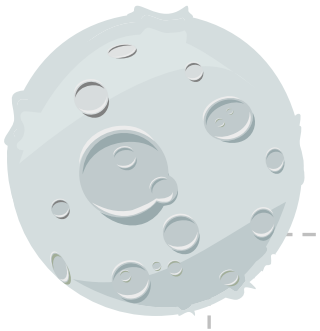
La Segunda Forma

The Second Way: Los principios del Feedback

"El objetivo de casi cualquier iniciativa de mejora de procesos es acortar y simplificar los bucles de retroalimentación para que se puedan hacer las correcciones necesarias de manera continua".



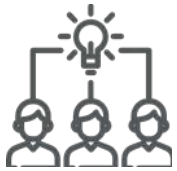
Competencias Necesarias – Capabilities



02 Ver los problemas a medida que ocurren

01 Trabajar de manera segura con sistemas complejos

03 "Swarm" para resolver problemas y construir nuevos conocimientos



04 Seguir presionando la calidad más cerca de la fuente

05 Habilitar la optimización para centros de trabajo downstream

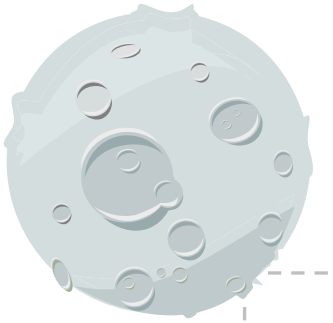
La Tercera Forma

The Third Way: Experimentación y Aprendizaje Continuos

"Crear una cultura que fomente dos cosas: la experimentación continua, que requiere tomar riesgos y aprender del éxito y el fracaso; y comprender que la repetición y la práctica es un requisito previo para el dominio".

"Mientras que First Way aborda el flujo de trabajo de izquierda a derecha y la Second Way aborda la respuesta recíproca rápida y constante de derecha a izquierda, la Third Way se enfoca en crear una cultura de aprendizaje y experiencia continuos. Estos son los principios que permiten la creación constante de conocimiento individual, que luego se convierte en conocimiento de equipo y organización" - Devops Handbook.

Competencias Necesarias – Capabilities



02 Institucionalizar la mejora del trabajo diario

03 Transformar los descubrimientos locales en mejoras globales

05 Los líderes refuerzan la cultura de aprendizaje

01 Habilitar el aprendizaje organizacional y una cultura segura

04 Inyectar patrones de resiliencia en nuestro trabajo diario

Trabajo en Equipos - Dinámica Flowban

Parte 3: La Primera Forma – FLUJO

3.0 Las Prácticas Técnicas del Flujo

¿Por qué la Primera Forma?

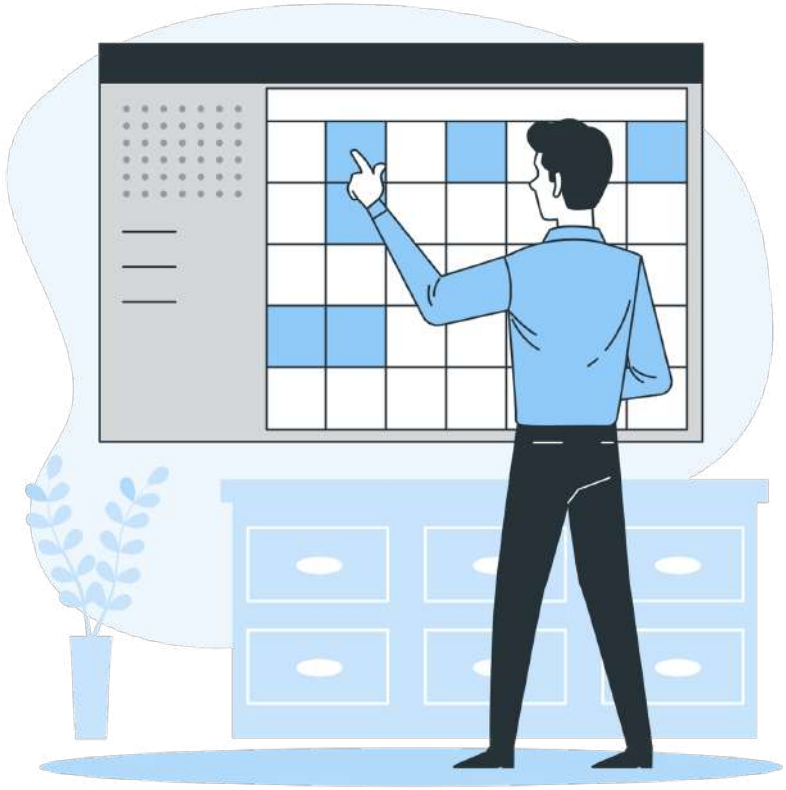
Nuestro objetivo es crear las prácticas técnicas y la arquitectura necesarias para permitir y sostener el rápido flujo de trabajo desde Desarrollo a Operaciones sin causar caos e interrupciones en el entorno de producción o en nuestros clientes. Esto significa que tenemos que reducir el riesgo asociado al despliegue y la liberación de los cambios en la producción. Para ello, aplicaremos un conjunto de prácticas técnicas conocidas como entrega continua. Los enfoques principales a ser cubiertos son:

- La creación de la base de nuestro pipeline de despliegue
- Habilitación de pruebas automatizadas rápidas y fiables
- Habilitar y practicar la integración y las pruebas continuas
- Automatización, habilitación y arquitectura para lanzamientos de bajo riesgo

Estas prácticas reducen el tiempo de espera para obtener entornos similares a los de producción, permite la realización de pruebas continuas con una rápida retroalimentación sobre el trabajo realizado, permite a los equipo desarrollar, probar y desplegar su código en producción de forma segura e independiente, y hace que los despliegues y lanzamientos de producción sean una parte rutinaria del trabajo diario.

Agenda

- 3.1 La creación de la base de nuestro pipeline de despliegue
- 3.2 Habilitación de pruebas automatizadas rápidas y fiables
- 3.3 Desarrollo Basado en Hipótesis
- 3.4 Habilitar y practicar la integración y las pruebas continuas
- 3.5 Automatización, habilitación y arquitectura para lanzamientos de bajo riesgo
- 3.6 Arquitectura para lanzamientos de Bajo riesgo



3.1 Las Prácticas Técnicas Pipeline de Despliegue

Crear los cimientos de nuestra línea de despliegue

Debatir cómo construir los mecanismos que nos permitirán crear entornos bajo demanda, ampliar el uso del control de versiones a todos los integrantes del flujo de valor, hacer que la infraestructura sea más fácil de reconstruir que de reparar y garantizar que los desarrolladores ejecuten su código en entornos similares a los de producción a lo largo de cada etapa del ciclo de vida del desarrollo de software.

En esta sección más adelante explicaremos:

1. Permitir la creación bajo demanda de entornos de desarrollo, prueba y producción
2. Crear nuestro único repositorio de la verdad para todo el sistema
3. Hacer que la infraestructura sea más fácil de reconstruir que de reparar
4. Modificar nuestra definición de desarrollo "hecho" para incluir la ejecución en entornos similares a los de producción

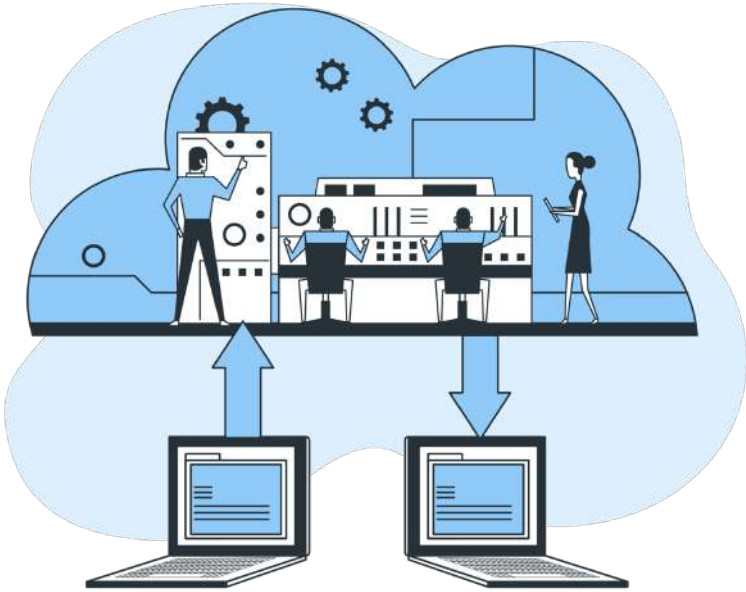
Flujo

Introducción

Debemos crear un flujo rápido y confiable de Dev a Ops para garantizar que siempre usamos entornos de producción en cualquier estado del flujo de valores.

Estos entornos deben ser creados de forma automatizada, idealmente bajo demanda desde scripts e información de configuración almacenados en control de versiones y totalmente autónomo, sin ningún trabajo manual requerido de Operaciones.

El objetivo es garantizar que podamos recrear todo el entorno de producción basado en el control de versiones.



El Mapeo del Flujo de Proceso es el punto de partida para la empresa que desea elaborar un plan bien estructurado para mejorar la productividad, rentabilidad, calidad, reducción de desperdicios y reducción de lead time.

3.1 El Pipeline de Despliegue

Pipeline de despliegue

- Cómo reducir el desperdicio
- Optimice el flujo de valor
- Controles de versión compartida
- Adaptación “Definición de Hecho”
- Automatice la Construcción y Configuración de entornos

Se crea una instancia del pipeline de despliegue cada vez que se realiza un cambio en una aplicación y permite los siguientes puntos:

- ✓ Reduce el tiempo de espera para obtener entornos de producción
- ✓ Permite pruebas continuas que dan a todos la respuesta rápida sobre su trabajo
- ✓ Permite que pequeños equipos desarrollan, prueban y despliegan con seguridad independientemente el código en producción
- ✓ Hace implementaciones de producción y libera una parte rutinaria del trabajo diario

3.1 El Pipeline de Despliegue

2.1.1 Evaluación de técnicas, infraestructura cómo código y contenedores

Abordaremos cómo crear mecanismos que nos permitan:

- Crear entornos bajo demanda
- Ampliar el uso del control de versiones para todos en el flujo de valor
- Hacer que la infraestructura sea más fácil de reconstruir que reparar
- Garantizar que los desarrolladores ejecuten su código en entornos de producción a lo largo de cada etapa del ciclo de vida del desarrollo de software

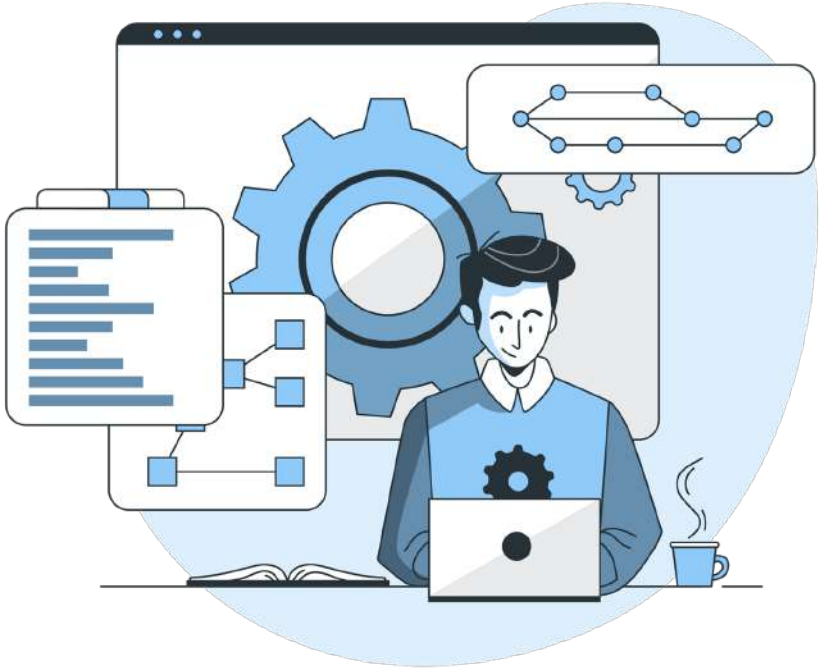
El objetivo de una pipeline es automatizar el proceso de entrega de software en producción de forma rápida, al mismo tiempo que garantiza su estabilidad, calidad y resiliencia.

3.1 El Pipeline de Despliegue

Infraestructura como código

Es el enfoque del proceso de gestión y aprovisionamiento a través de la lectura de archivos de definición de máquina en lugar de configuración de hardware físico o herramientas de configuración interactiva.

Las configuraciones deben estar en un sistema de control de versiones. Puede utilizar secuencias de comandos o definiciones declarativas en lugar de procesos manuales.

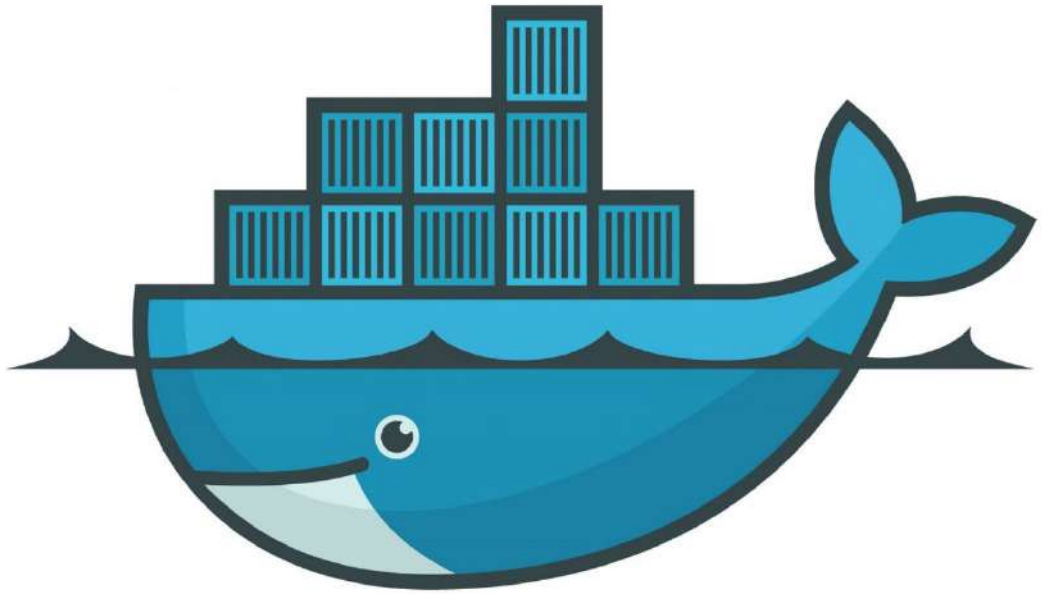


3.1 El Pipeline de Despliegue

Contenedores

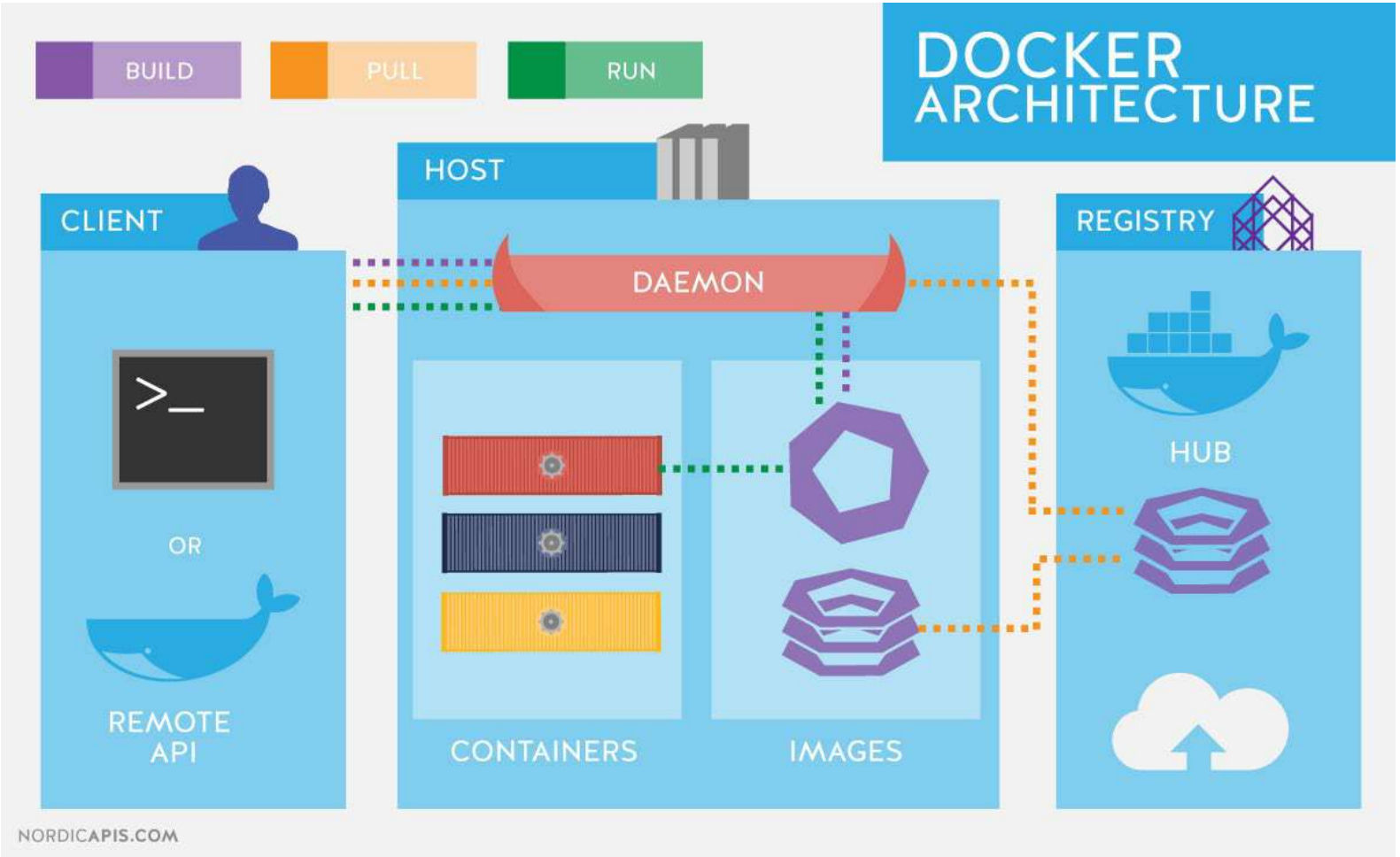
Proporcionan una alternativa ligera a las máquinas virtuales y permiten a los desarrolladores trabajar con entornos y pila de DEV en forma idéntica.

Contenerización es una alternativa ligera a la virtualización completa de la máquina que implica encapsular una aplicación en un contenedor con su propio entorno operativo.



Docker Architecture

Fuente: https://compbcn.es/docker-en-pocas-palabras/



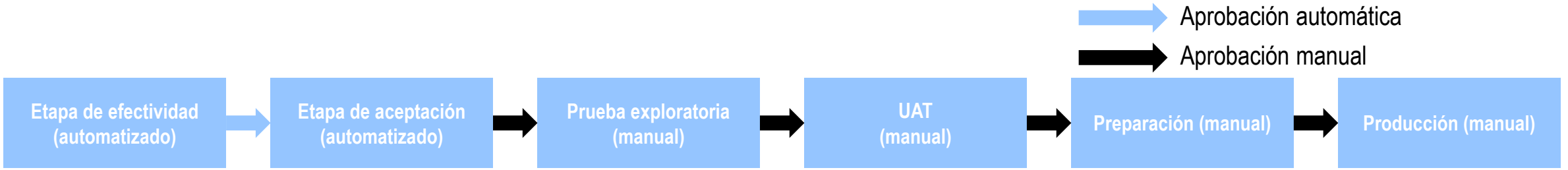
3.1 El Pipeline de Despliegue

Al definir cuidadosamente todos los aspectos del entorno antes del tiempo, es posible crear nuevos entornos rápidamente, y garantizar que estos entornos sean estables, confiables, consistentes y seguros.

- La **Operación se beneficia** al crear nuevos entornos rápidamente, refuerza la consistencia y reduce el trabajo manual tedioso y propenso a errores
- El **Desarrollo se beneficia** al reproducir todas las partes necesarias del entorno de producción para crear, ejecutar y probar su código en sus estaciones de trabajo
- El **Desarrollo puede** reproducir, diagnosticar y corregir defectos rápidamente, aislados con seguridad de servicios de producción y otros recursos compartidos
- El **Desarrollo puede** experimentar cambios en los entornos, así como la infraestructura como un código, creando aún más conocimiento compartido entre Desarrollo y Operaciones

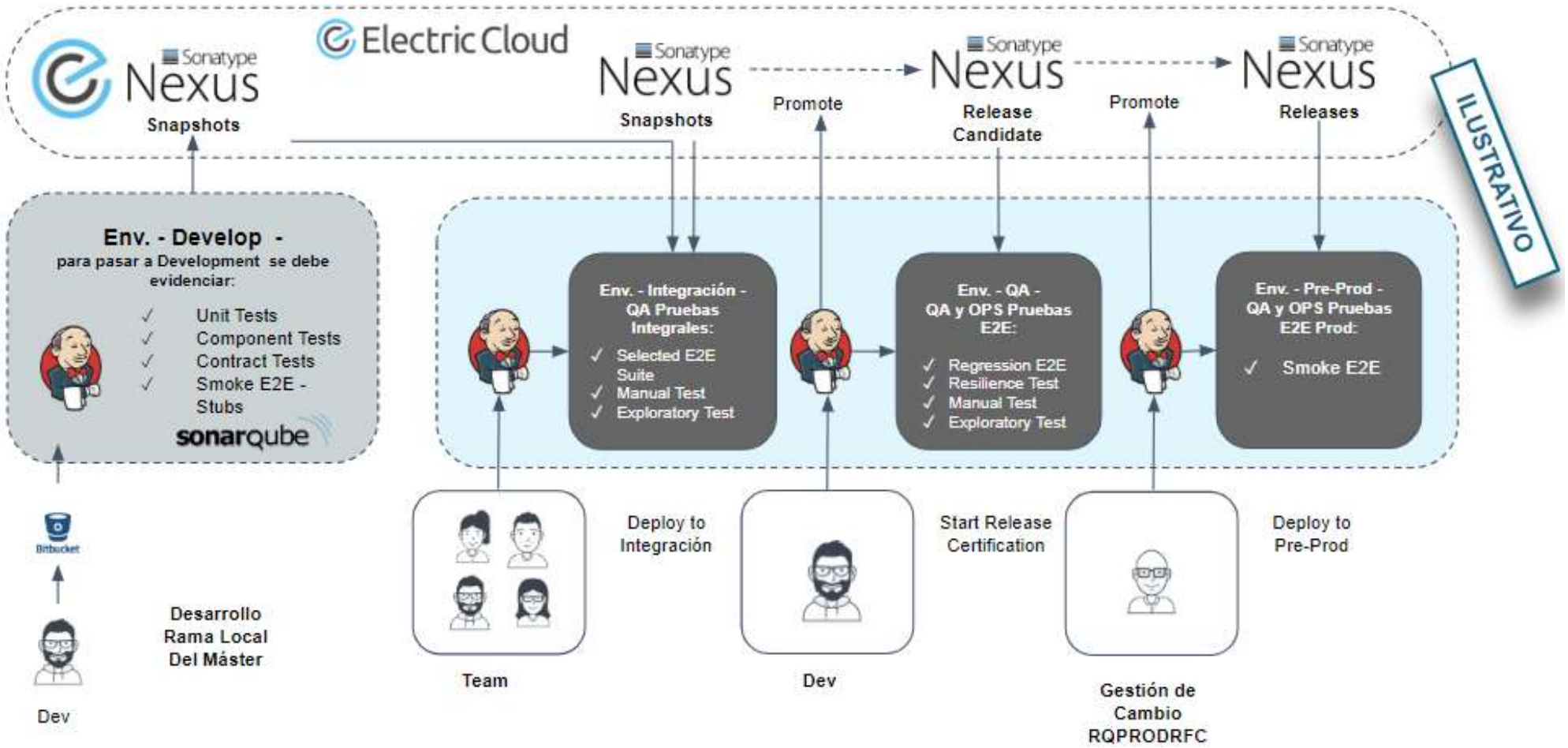
3.1 El Pipeline de Despliegue

2.1.2 Mejor solución para optimizar el flujo de valor



- Definición de "Hecho"
- Cualquier provisión de entornos similares al de producción bajo demanda
- Reducir el riesgo de producción
- Las operaciones pueden hacer que los desarrolladores sean mucho más productivos
- Todos los artefactos de producción en el control de versiones. Hay una "única fuente de la verdad“
- Infraestructura de producción con foco más en reconstruir que reparar

Principios de DevOps



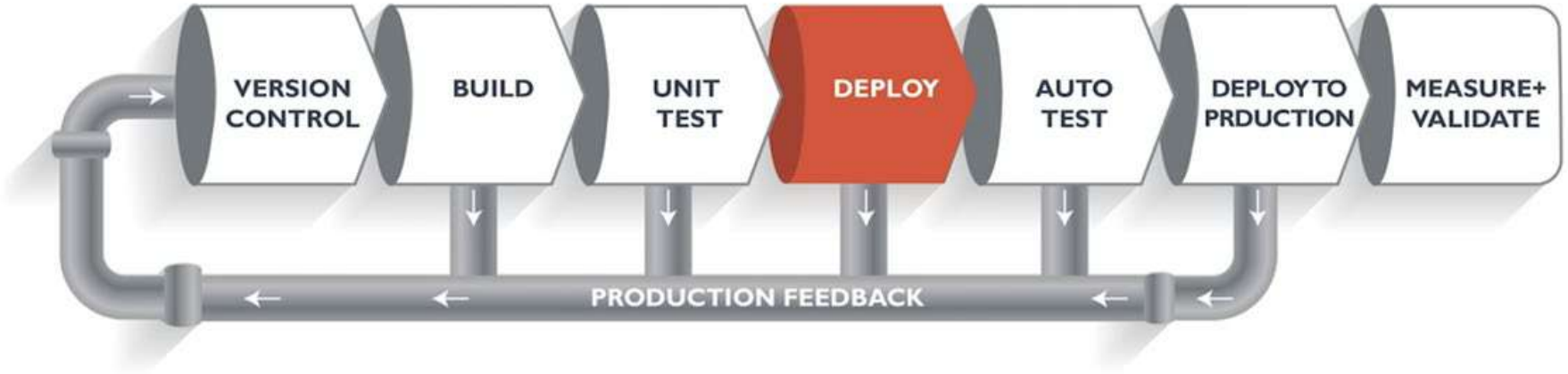
Principios de DevOps

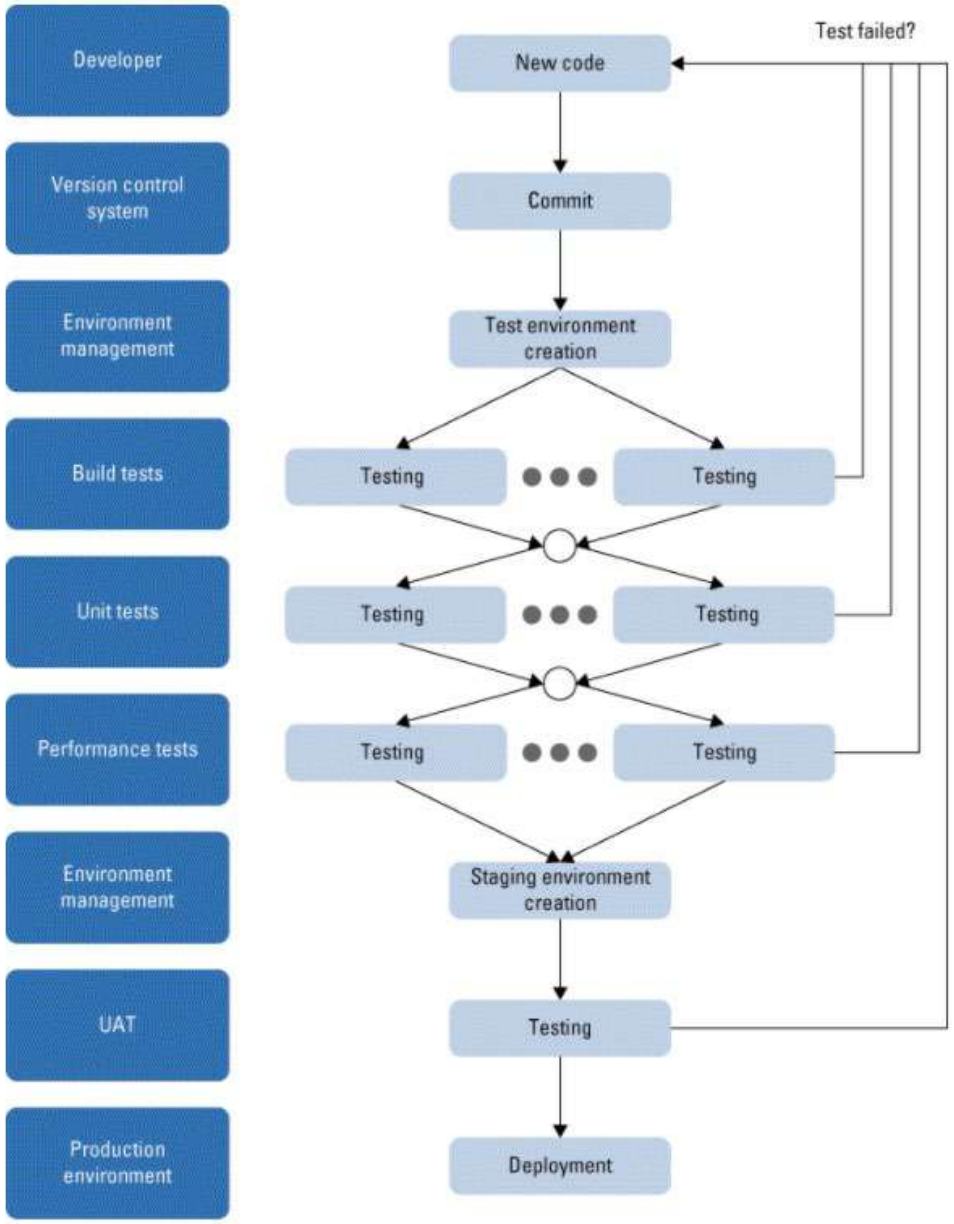
Pipeline de Despliegue

Se usa el deployment pipeline para ayudar a la revisión de la cadena de valor, esto es la transición más automatizada de los cambios a través de todos los pasos de la cadena de valor, comenzando desde el punto 'El desarrollo está completo', hasta 'Implementado en operaciones'.

El pipeline ayuda a lidiar con tareas importantes de DevOps. Primero, ahorra recursos al no comenzar los siguientes pasos antes de que se completen los anteriores. Segundo, garantiza la calidad del producto: los cambios que no funcionan según lo requerido, no alcanzan el entorno de producción. Tercero, acelera la entrega de cambios en el entorno de producción al maximizar la automatización de cada paso. Y cuarto, constantemente deja registros en los logs de auditoría, lo que proporciona datos valiosos para su optimización.

Principios de DevOps





Principios de DevOps

Ayuda a lidiar con 4 tareas de DevOps:

- Ahorra recursos al no empezar una etapa antes de finalizar la otra
- Asegura la calidad del producto, ya que los cambios que no se comporten como lo esperado no alcanzan producción
- Acelera la entrega de cambios a producción automatizando cada paso
- Deja registros y logs constantes lo que permite el monitoreo de los cambios realizados y permiten la medición de cada etapa proveyendo data para la optimización

3.1 El Pipeline de Despliegue

2.1.3 Repositorio de control de versión compartida para la integridad

Todos los archivos y configuraciones de su aplicación deben estar en control de versiones; se convierte en el único repositorio de confianza que contiene el estado deseado y preciso del sistema.

Permite reproducir repetidamente y de forma confiable todos los componentes de nuestro sistema de software de trabajo, que incluye nuestras aplicaciones y entorno de producción, así como todos nuestros entornos de preproducción.

- Todos los códigos y dependencias de la aplicación
- Cualquier script utilizado para crear esquemas de base de datos, referencia de aplicación de datos, etc
- Todas las herramientas de creación de entorno y artefactos descritos en el paso anterior
- Cualquier archivo utilizado para crear contenedores
- Todas las pruebas automatizadas de soporte y cualquier script de prueba manual
- Cualquier script que admita paquetes de código, implementación, migración de base de datos y aprovisionamiento de entorno
- Todos los artefactos del proyecto
- Todos los archivos de configuración de la nube
- Cualquier otro script o información de configuración necesaria para crear infraestructura que soporta varios servicios

3.1 El Pipeline de Despliegue

2.1.4. Definición de Hecho (DoD) para DevOps

El objetivo es asegurar que el Desarrollo y el Control de Calidad estén integrando rutinariamente el código con entornos similares al de producción a intervalos cada vez más frecuentes a lo largo del proyecto.

Si se expande la definición de "Hecho" además de sólo la funcionalidad de código correcta, al final de cada intervalo de desarrollo, se integra, prueba, se trabaja con el potencial despliegue del código, demostrados en un entorno similar al de producción.

3.1 El Pipeline de Despliegue

2.1.5 Herramientas se pueden utilizar para automatizar la construcción y la configuración del entornos

Una de las principales causas contributivas de las implementaciones de software caótico, disruptivo y catastrófico, es la primera vez que la aplicación se comporte en el entorno de producción con un conjunto de datos reales durante la liberación.

- En algunos casos, los equipos de desarrollo pueden haber solicitado entornos de prueba en las etapas iniciales del proyecto
- Largos tiempos de espera (lead time) para el aprovisionamiento de entornos de pruebas
- Entornos con falta de datos adecuados
- Entornos de prueba mal configurados o diferentes de la operación

3.1 El Pipeline de Despliegue

Queremos que los desarrolladores ejecuten sus códigos en entornos similares al de producción en sus propias estaciones de trabajo, creadas bajo demanda y en servicios de autoservicio.

- Ofrecer un mecanismo de provisión, que crea todos nuestros entornos, como desarrollo, prueba y producción

Para ello, es necesario definir y automatizar la creación de nuestros entornos conocidos y buenos, de manera estable, segura y de riesgo reducido.
Los requerimientos deben ser incorporados en el proceso automatizado de construcción del entorno.

3.1 El Pipeline de Despliegue

Usar la automatización para cualquiera o todos los siguientes (en lugar de construcciones manuales de entornos y sus configuraciones):

- Copiando un entorno virtualizado
- Construcción automatizada de entorno “metal crudo”
- Uso de herramientas de administración de configuración ("infraestructura como código")
- Uso de herramientas automáticas de configuración del sistema operativo
- Montar un entorno desde un conjunto de imágenes o contenedores virtuales
- Subir un nuevo entorno en una nube pública, nube privada u otras PaaS

Talleres Prácticos Sugeridos

3.2 Las Prácticas Técnicas Pruebas Automatizadas

Permitir pruebas automatizadas rápidas y fiables (1 / 2)

Repasar las prácticas de integración continua necesarias para crear las prácticas de pruebas automatizadas que garanticen que los desarrolladores obtengan rápidamente información sobre la calidad de su trabajo. Esto es aún más importante a medida que aumentamos el número de desarrolladores y de ramas en las que trabajan en el control de versiones.

En esta sección explicaremos:

- Construir, probar e integrar continuamente nuestro código y entornos
- Construir un conjunto de pruebas de validación automatizadas rápido y fiable
- Detectar los errores lo antes posible en nuestras pruebas automatizadas
- Garantizar que las pruebas se ejecuten rápidamente (en paralelo, si es necesario)
- Escribir nuestras pruebas automatizadas antes de escribir el código ("desarrollo dirigido por pruebas")

3.2 Las Prácticas Técnicas Pruebas Automatizadas

Permitir pruebas automatizadas rápidas y fiables (2 / 2)

En esta sección explicaremos :

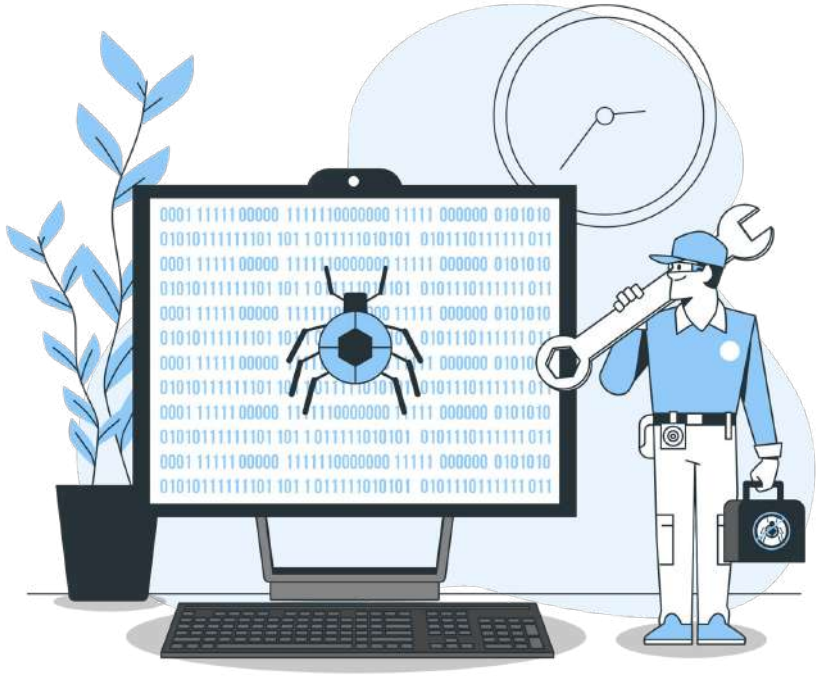
- Automatizar tantas pruebas manuales como sea posible
- Integrar las pruebas de rendimiento en nuestro conjunto de pruebas
- Integrar las pruebas de requisitos no funcionales en nuestro conjunto de pruebas
- Tirar de la cuerda de andon cuando se rompa la tubería de despliegue
- Por qué tenemos que tirar de la cuerda de andon

3.2 Pruebas Automatizadas

Las pruebas automatizadas abordan otro problema significativo e inquietante.

Sin pruebas automatizadas, cuanto más código se escribe, más tiempo y dinero son necesarios para probar el código, y en la mayoría de los casos, es un modelo no escalable para cualquier organización de tecnología.

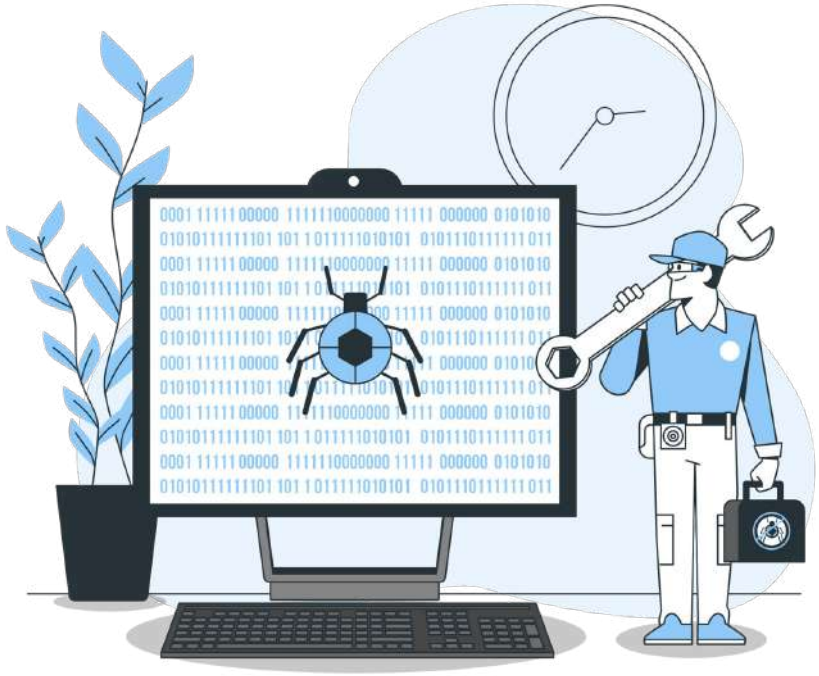
Con la automatización de pruebas, es posible evitar las actividades manuales y repetitivas que sobrecargan tanto el presupuesto como el cronograma de una empresa. Además, todavía existe la posibilidad de crear pruebas más amplias, elaboradas y que estén de acuerdo con las funcionalidades y exigencias de sus productos.



3.2 Pruebas Automatizadas

Las pruebas en DevOps se pueden dividir de la siguiente manera:

- **Unitarios:** permiten la reducción de las unidades de las clases, aplicaciones y validaciones de tamaño de campos
- **Integrados:** fomentan la integración de las aplicaciones existentes en los sistemas
- **Visuales:** aseguran el funcionamiento del layout y de los elementos estáticos de una aplicación
- **Funcionales:** garantizan un buen desempeño de las funcionalidades
- **Performance:** hacen que los releases atiendan a las especificaciones previamente definidas y los comparan con resultados anteriores



3.2 Pruebas Automatizadas

Crear calidad en el producto, desde las etapas iniciales con pruebas automatizadas en el trabajo diario del desarrollador.

Así, se crea un loop de feedback rápido que ayuda a los desarrolladores a encontrar y corregir problemas rápidamente, cuando hay menos restricciones (por ejemplo, tiempo, recursos).

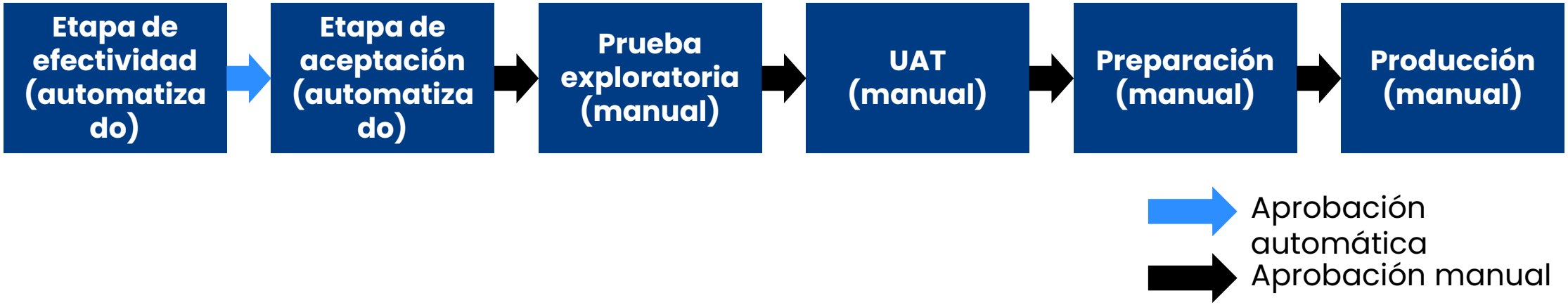
Los procesos automatizados de construcción y prueba se vuelven críticos por los siguientes motivos:

- El proceso de construcción y prueba se puede ejecutar todo el tiempo
- Entender todas las dependencias necesarias para construir, empaquetar, ejecutar y probar nuestro código
- El empaquetado de la aplicación permite la instalación repetitiva de código y configuraciones en un entorno
- Se puede optar por empaquetar nuestras aplicaciones en contenedores
- Los entornos pueden volverse más parecidos a la producción de una manera consistente y repetible

3.2 Pruebas Automatizadas

El pipeline de despliegue válida después de cada cambio, que el código se integra con éxito en un entorno de producción.

Se convierte en la plataforma a través de la cual los analistas de pruebas solicitan y certifican compilaciones durante pruebas de aceptación y pruebas de usabilidad, y donde ejecutarán validaciones automatizadas de rendimiento y seguridad.



3.2 Pruebas Automatizadas

En general, las pruebas automatizadas se encuadran en una de las siguientes categorías, desde el más sencillo al más complejo de implementar:

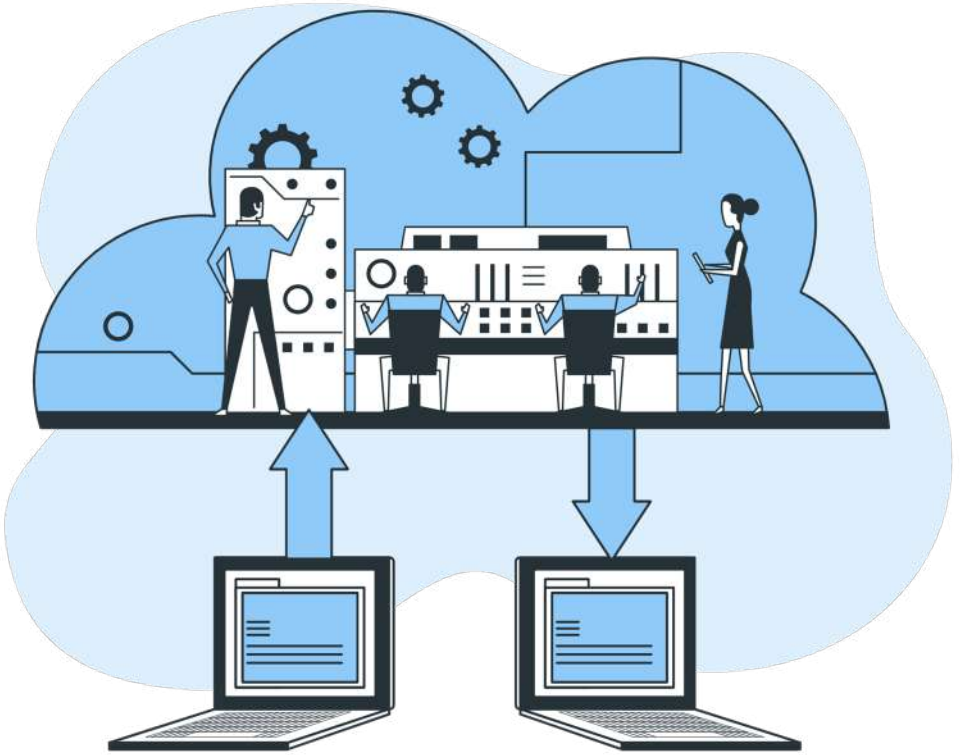
- Pruebas de unitarias
- Pruebas de integración
- Pruebas de aceptación

3.2 Pruebas Automatizadas

Cualquier error debe ser encontrado lo más temprano posible.

- ✓ Si la mayoría de nuestros errores se encuentran en nuestras pruebas de aceptación e integración, el feedback que proporcionamos a los desarrolladores es mucho más lento que en las pruebas unitarias

Por lo tanto, siempre que encontremos un error con una prueba de aceptación o integración, debemos crear unas pruebas unitarias que puedan encontrar el error más rápido, más temprano y más barato.

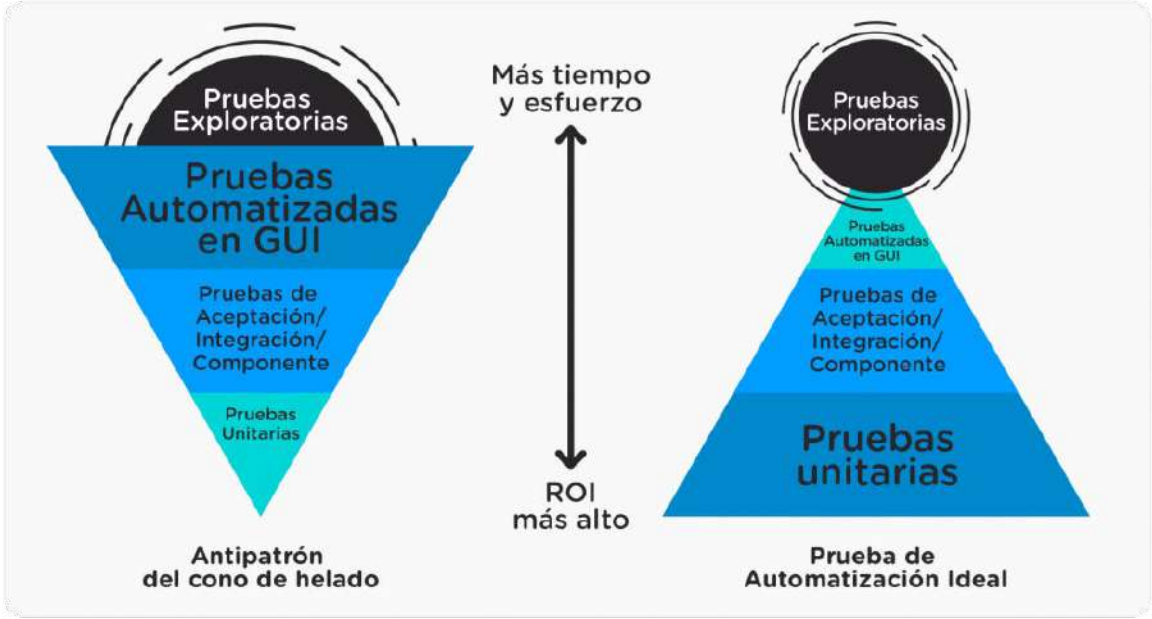


3.2 Pruebas Automatizadas

Una señal que tenemos una arquitectura fuertemente acoplada:

- Muchas veces tenemos consecuencias inesperadas en otros módulos diferentes al módulo que estamos desplegando
- Escribir y mantener pruebas de unitarias y de aceptación es difícil y costoso

En ese caso, necesitaremos crear un sistema más débilmente acoplado para que los módulos puedan ser probados independientemente.



3.2 Pruebas Automatizadas

Una de las maneras más eficaces para las pruebas automatizadas confiables es utilizar técnicas como:

- Desarrollo orientado a pruebas (TDD)
- Desarrollo orientado a pruebas de aceptación (ATDD)

Es cuando empezamos todos los cambios en el sistema, primero escribiendo una prueba automatizada que valida el comportamiento esperado y después escribimos el código que pasará por las pruebas.

Kent Beck (1990) en Extreme Programming, define tres etapas:

1. **Asegúrese de que las pruebas fallen**. "Escriba una prueba para el siguiente bit de funcionalidad que desea agregar". Check-in
2. **Asegúrese de que las pruebas sean satisfactorias**. "Escriba el código funcional hasta que la prueba pase". Check-in
3. **Refactorizar el código nuevo y antiguo para que esté bien estructurado**. Asegúrese de que las pruebas sean satisfactorias. Check-in de nuevo

Talleres Prácticos Sugeridos

3.3 Desarrollo Orientado por Pruebas

Automatice tantas Pruebas Manuales como sea posible

- ✓ Aunque las pruebas se pueden automatizar, la creación de calidad no puede. Tener personas que realizan pruebas que se deben automatizar es un desperdicio de potencial humano
- ✓ Así, habilitamos a todos nuestros analistas de pruebas (lo que, por supuesto, incluye desarrolladores)
- ✓ En trabajos de actividades de alto valor que no se pueden automatizar, cómo probar o mejorar el proceso de prueba en sí
- ✓ Un pequeño número de pruebas confiables y automatizadas es casi siempre preferible a un gran número de pruebas manuales automáticas o no confiables
- ✓ Comenzamos con un pequeño número de pruebas automatizadas confiables y añadimos a ellos a lo largo del tiempo

Desafíos de automatización de pruebas:

1. Arquitectura de automatización de pruebas
2. Paradigmas de automatización de pruebas
3. Costo de la automatización y mantenimiento de las pruebas
4. Profesionales calificados
5. Entorno de prueba
6. La garantía de la calidad
7. Expectativa de que el retorno de inversión en automatización sea de corto plazo

3.3 Desarrollo Orientado por Pruebas

Integre las pruebas de performance en su grupo de pruebas

El objetivo es crear y ejecutar **pruebas de performance automatizadas** que validen la performance en toda la pila de aplicaciones (código, base de datos, almacenamiento, red, virtualización, etc.) como parte del pipeline de despliegue para detectar problemas precozmente, cuando las correcciones son más baratas y más rápidas.

Las aplicaciones y los entornos se comportan bajo una carga de producción, podemos hacer un trabajo mucho mejor en la planificación de la capacidad, así como detectar condiciones como:

- Cuando los tiempos de consulta de base de datos crecen de forma no lineal (por ejemplo, olvidamos de activar la indexación de la base de datos y la carga de la página pasa de cien milisegundos a treinta segundos)
- Cuando un cambio de código hace que el número de llamadas de base de datos, uso de almacenamiento o tráfico de red aumenta diez veces

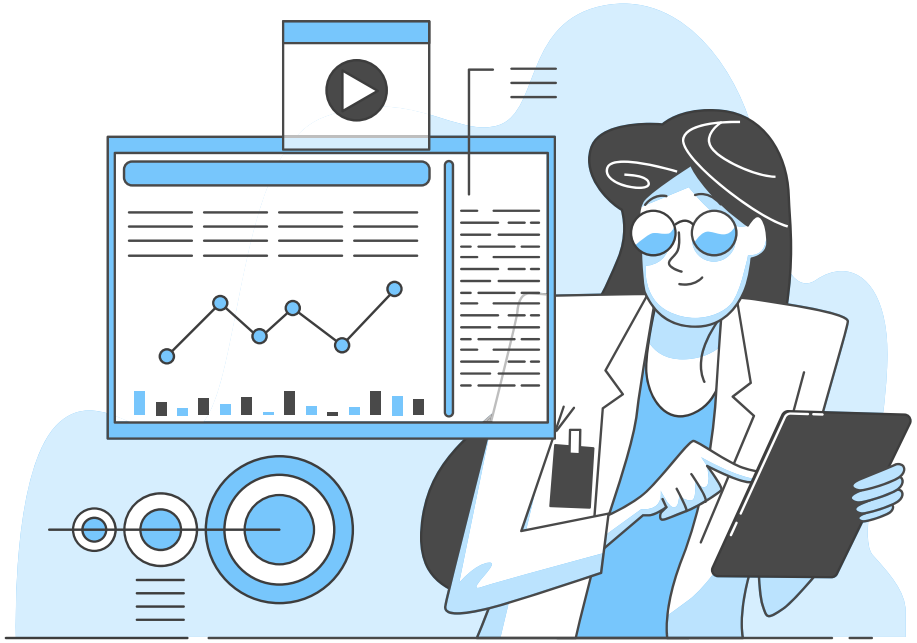
3.3 Desarrollo Orientado por Pruebas

Integre pruebas no funcionales en su grupo de pruebas

Es necesario validar todos los demás atributos relevantes del sistema, llamados requisitos no funcionales, que incluyen:

- Disponibilidad
- Escalabilidad
- Capacidad
- Seguridad

...y así sucesivamente.



3.3 Desarrollo Orientado por Pruebas

Tire de la cuerda ANDON cuando el pipeline de despliegue se rompe

Para mantener el pipeline de despliegue en un estado verde, crearemos una cuerda Andon virtual, similar al físico en el Sistema Toyota de Producción.

Siempre que alguien introduzca un cambio que hace que nuestra creación o pruebas automatizadas fallen, ningún nuevo trabajo puede entrar en el sistema hasta que el problema se corrija. Y si alguien necesita ayuda para resolver el problema, puede traer la ayuda que necesite.

Discusión en Grupos

3.4 Integración Continua

Habilitar y practicar la integración continua

Tras el uso exhaustivo del control de versiones, la integración continua es una de las prácticas más críticas que permiten el rápido flujo de trabajo en nuestro flujo de valor, permitiendo a muchos equipos de desarrollo desarrollar, probar y entregar valor de forma independiente.

En esta sección explicaremos:

1. El desarrollo en lotes pequeños y lo que sucede cuando confirmamos el código al tronco con poca frecuencia
2. Adoptar prácticas de desarrollo basadas en el tronco

3.4 Integración Continua

Trabajar en BRANCH

Desarrollo en BRANCH en control de versiones: Se creó principalmente para permitir a los desarrolladores trabajar en diferentes partes del sistema de software en paralelo, sin el riesgo de que los desarrolladores individuales **verificaron** cambios que podrían desestabilizar el trabajo o incluso de introducir errores en el TRUNK (o maestro o mainline).

Desventajas:

- Más esfuerzo en BRANCH, más difícil integrar y combinar los cambios de todos en el tronco
- Integrar estos cambios se vuelve exponencialmente más difícil
- Los problemas de integración resultan en una cantidad significativa de retrabajo
- Si se realiza al final del proyecto (tradicionalmente es así), toma mucho más tiempo
- Espiral descendente: cuando la fusión de código es "dolorosa", tendemos a hacerlo con menos frecuencia haciendo las futuras mezclas aún peores

3.4 Integración Continua

La integración continua fue diseñada para resolver este problema, haciendo que la combinación en el trunk una parte del trabajo diario de todos.

La integración continua resuelve una variedad sorprendente de problemas. Los principales objetivos de la integración continua son encontrar e investigar bugs más rápidamente, mejorar la calidad del software y reducir el tiempo que tarda en validar y lanzar nuevas actualizaciones de software.

"Sin pruebas automatizadas, la integración continua es la manera más rápida de obtener una gran pila de basura electrónica que nunca se compila o se ejecuta correctamente".

La integración continua es una práctica de desarrollo de software de DevOps en la que los desarrolladores, a menudo, juntan sus cambios de código en un repositorio central. Después de esto, se ejecutan creaciones y pruebas.

3.4 Integración Continua

Gary Gruver, director de ingeniería de la división HP LaserJet Firmware, donde crean el software embarcado para todos sus escáneres, impresoras y dispositivos multifunción.

El equipo estaba compuesto por cuatrocientos desarrolladores distribuidos en Estados Unidos, Brasil e India. A pesar del tamaño del equipo, se estaban moviendo muy lentamente. Durante años, no pudieron proporcionar nuevos recursos tan rápidamente como los negocios necesitaban.

Gruver describió: "El marketing llegaba a nosotros con un millón de ideas para deslumbrar a nuestros clientes y les decíamos: 'Quítate de tu lista, elige las dos cosas que te gustaría obtener en los próximos seis o doce meses".

3.4 Integración Continua

Sólo estaban concluyendo dos lanzamientos al año, con la mayor parte del tiempo dedicado a la portabilidad de código para soportar nuevos productos. Gruver estimó que sólo el 5% de su tiempo se gastó en la creación de nuevos recursos - el resto del tiempo fue trabajo no productivo asociado a la deuda técnica, como la gestión de varias ramas de código y pruebas manuales, como se muestra a continuación:

- 20 % en la planificación detallada (su baja productividad y largos períodos de tiempo se asignaron erróneamente a una estimación defectuosa y, en espera de una respuesta mejor, se pidió más detalles)
- 25 % en la portabilidad de código, todos mantenidos en branches de código separados
- 10 % en la integración del código entre branch de desarrollador
- 15 % en el llenado de pruebas manuales

3.4 Integración Continua

Gruver y su equipo crearon una meta de acelerar el tiempo gastado en innovación y nuevas funcionalidades por un factor de diez. El equipo esperaba que ese objetivo pudiera ser alcanzado por medio de:

- Integración continua y desarrollo basado en TRUNK
- Inversión significativa en la automatización de pruebas
- Creación de un simulador de hardware para que las pruebas puedan ejecutarse en una plataforma virtual
- La reproducción de errores de prueba en las estaciones de trabajo del desarrollador
- Una nueva arquitectura para soportar la ejecución de todas las impresoras de un build y release comunes

3.4 Integración Continua

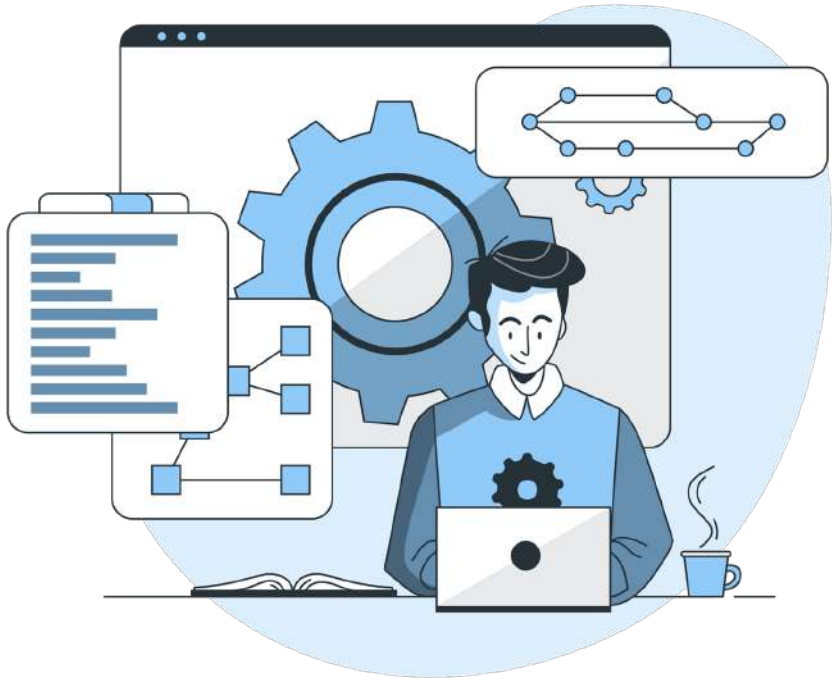
Las estrategias de ramificación o de uso de BRANCH

1) Optimizar para la productividad individual:

Cada desarrollador trabaja en su propio branch privado.

Todos trabajan independientes y nadie obstaculiza el trabajo del otro; sin embargo, la fusión o mezcla se convierte en una pesadilla.

La colaboración se vuelve difícil - el trabajo de cada uno tiene que ser meticulosamente integrado con el trabajo de todos para ver hasta la parte más pequeña del sistema completo.



3.4 Integración Continua

2) Optimizar la productividad del equipo:

Todos trabajan en la misma área común. No hay branch, sólo un trunk de desarrollo largo e ininterrumpido.

- Verificar el código con frecuencia reduce el tamaño de lote para el trabajo
- Cuanto más registran el código en el tronco, más cerca del ideal teórico del flujo de pieza única
- Ejecutar todas las pruebas automatizadas y recibir alertas de cambios de interrupción
- Con la detección de problemas de mezcla aún pequeños, podemos corregirlos más rápidamente
- Método confirmaciones bloqueadas: sólo es posible realizar la confirmación en el pipeline si el cambio enviado se aprueba en todas las pruebas automatizadas antes de ser mezclada en el trunk
- El control de versiones se convierte en un mecanismo integral de cómo el equipo se comunica entre sí

3.4 Integración Continua

Adopte prácticas de desarrollo basadas en el TRUNK

Una medida importante para las grandes mezclas es instituir prácticas de integración continua y de desarrollo basado en TRUNK:

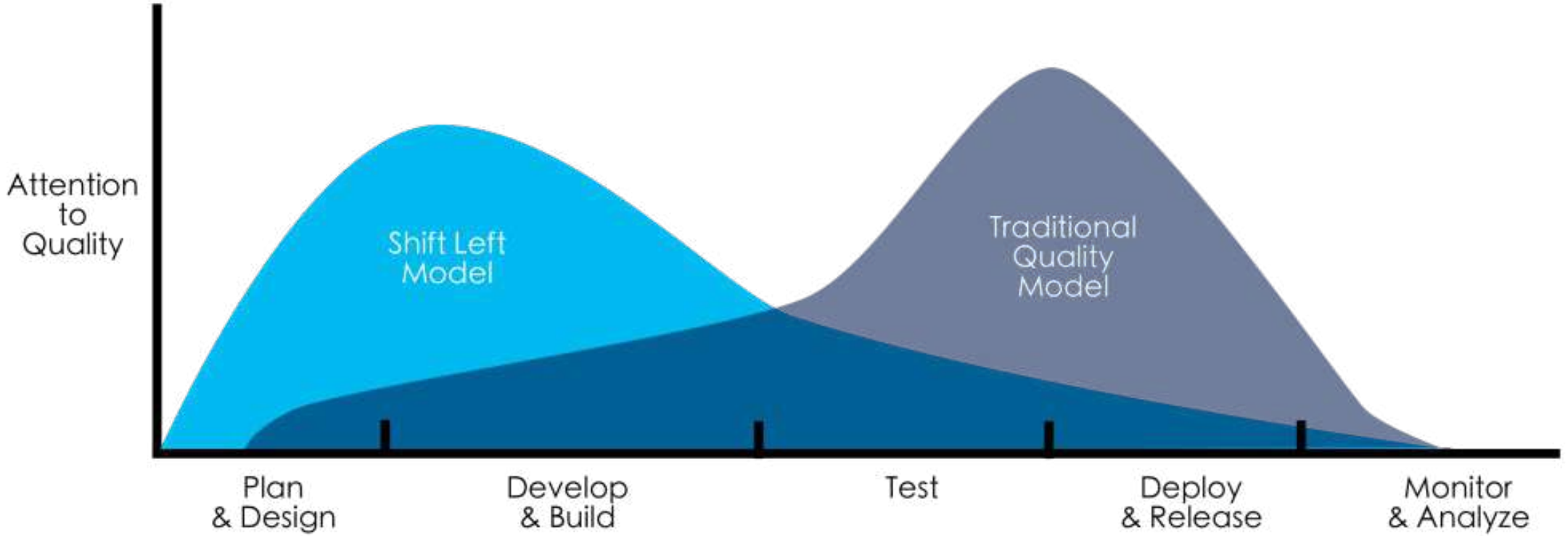
- Los desarrolladores registran su código en el tronco al menos una vez al día

Verificar el código a menudo reduce el tamaño del lote para el trabajo realizado por todo nuestro equipo de desarrolladores en un solo día.

Cuanto más los desarrolladores registran el código en el tronco, menor el tamaño del lote y más próximos estamos del ideal teórico del flujo de pieza única.

Definición de "listo": *"Al final de cada intervalo de desarrollo, debemos tener código integrado, probado, funcional y potencialmente utilizable, demostrado en un entorno similar a la producción, creado a partir del trunk usando un proceso de un clic y validado con pruebas automatizado".*

Sponsored By: Shift Left Concept



Fuente: https://jaxenter.com/devops-shifting-left-172792.html

3.4 Integración Continua

Automatice el proceso de Despliegue

- Documentar las etapas del proceso de Despliegue, en el mapeo del flujo de valor
- Simplifique y automatice el mayor número posible de etapas manuales, como:
  - ✓ Empaquetado del código en las formas adecuadas para la implementación
  - ✓ Creación de imágenes o contenedores de máquina virtual preconfigurados
  - ✓ Automatización de la implementación y configuración de middleware
  - ✓ Copiar paquetes o archivos a servidores de producción
  - ✓ Reiniciar servidores, aplicaciones o servicios
  - ✓ ....

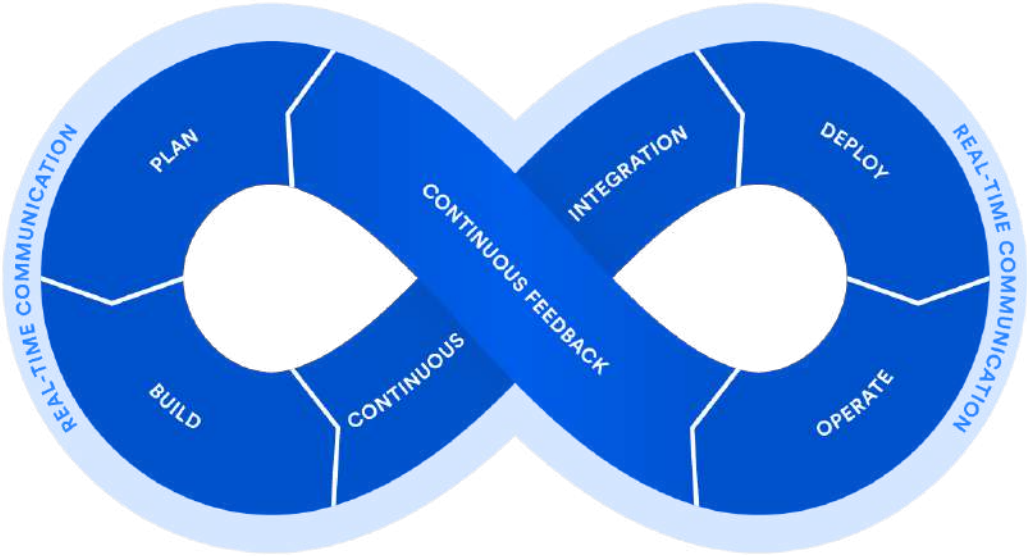
3.4 Integración Continua

Automatice el proceso de Despliegue

- Intente remodelar para eliminar etapas
- También intente reducir los plazos de entrega y el número de transferencias
- Involucre desarrolladores en la automatización y optimización del proceso de Despliegue
- Ponga el equipo de desarrollo a trabajar en estrecha colaboración con las Operaciones

3.4 Integración Continua

Automatice el proceso de Despliegue



Talleres Prácticos Sugeridos

3.5 Las Prácticas Técnicas Releases

Automatice y habilite las liberaciones de alto riesgo

En este capítulo, reducimos la fricción asociada a los despliegues de producción, asegurando que se puedan realizar con frecuencia y fácilmente, ya sea por Operaciones o por Desarrollo. Para ello, ampliaremos nuestro pipeline de despliegue.

En lugar de limitarse a integrar continuamente nuestro código en un entorno similar al de producción, habilitaremos la promoción a producción de cualquier compilación que supere nuestro proceso automatizado de pruebas y validación, ya sea bajo demanda (es decir, pulsando un botón) o automáticamente (es decir, cualquier compilación que supere todas las pruebas se despliega automáticamente).

3.5 Lanzamiento de Alto Riesgo

El principal tema que debemos preocuparnos no es la forma, sino los resultados: las implementaciones deben ser eventos de "presionar un botón" de bajo riesgo que podemos ejecutar bajo demanda.

- Trabajando en pequeños lotes en el TRUNK, y siempre el código se mantiene en un estado liberable, podemos liberar bajo demanda "presionando un botón" durante el horario comercial normal, estamos haciendo una entrega continua
- Implementando buenos builds en producción regularmente a través del Auto-Servicio - desplegando a producción **al menos una vez al día por desarrollador**, o automáticamente todos los cambios que un desarrollador completa - es cuando nos estamos comprometiendo al despliegue continuo

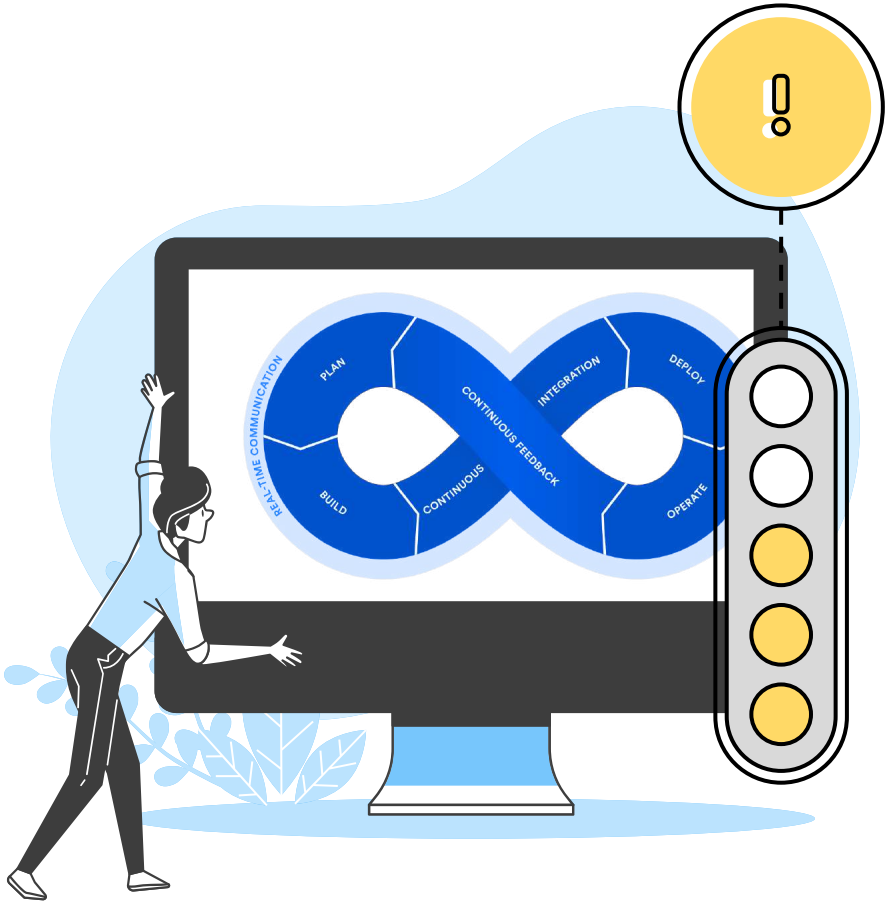
*Definida de esta manera, la **entrega continua** es el requisito previo para el despliegue continuo, así como la **integración continua** es un requisito previo para la **entrega continua**.*

3.5 Lanzamiento de Alto Riesgo

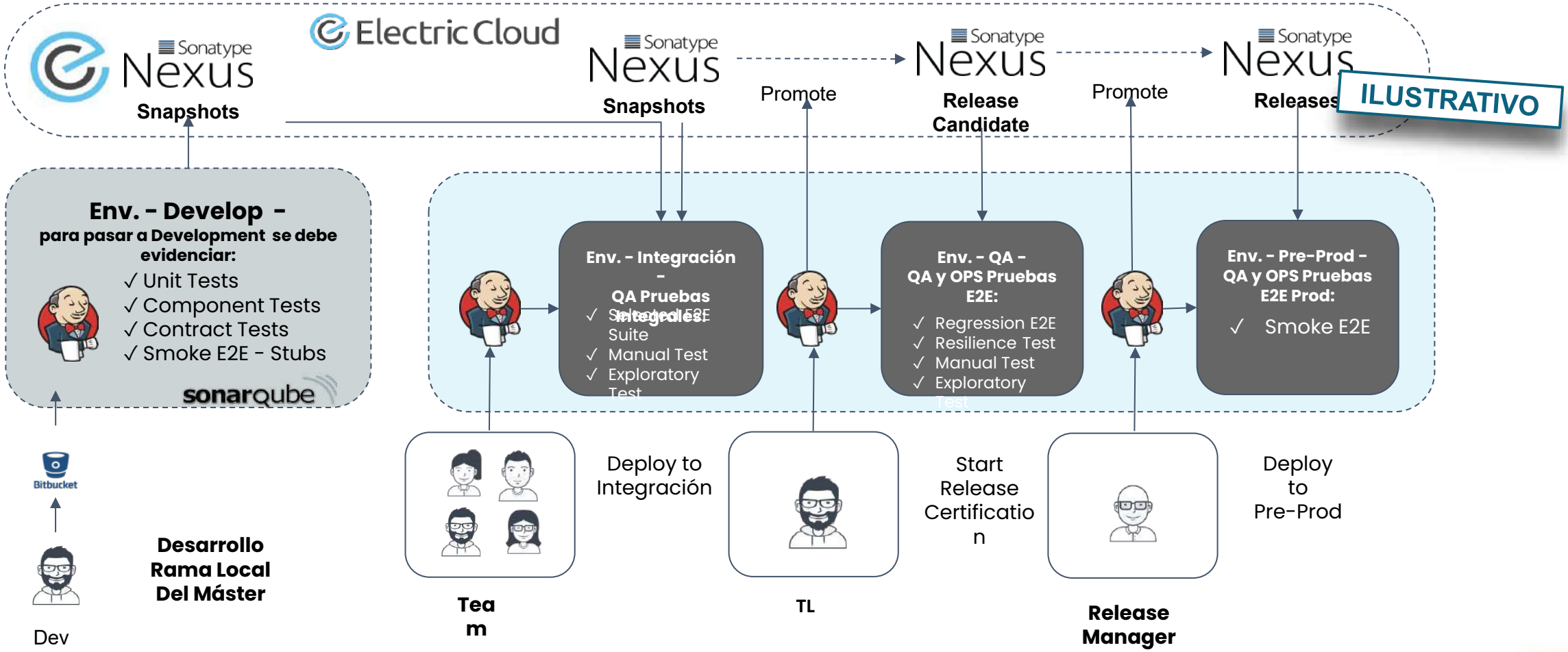
Automatice el proceso de Despliegue

Los requisitos para el pipeline de despliegue incluyen:

- ✓ Despliegue de la misma manera en todos los entornos
- ✓ Prueba de humo de nuestras implementaciones
- ✓ Garantizar el abastecimiento y mantenimiento de entornos consistentes (para Dev, QA, Soporte y Producción)



CI / CD Pipelines para Servicios



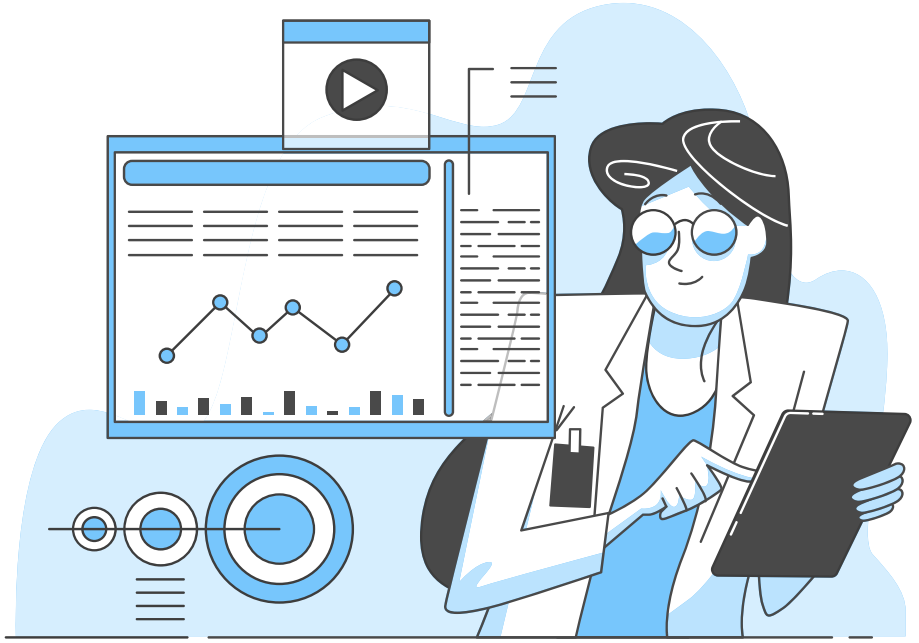
3.5 Lanzamiento de Alto Riesgo

Automatice el proceso de Despliegue

Proporciona implementaciones automatizadas en forma de autoservicio:

Para habilitar mejor el flujo rápido, deseamos un proceso de promoción de código que pueda ser ejecutado por Desarrollo o Operaciones, idealmente sin ninguna etapas manuales o transferencias. Esto afecta a los pasos siguientes:

- Build
- Prueba
- Despliegue

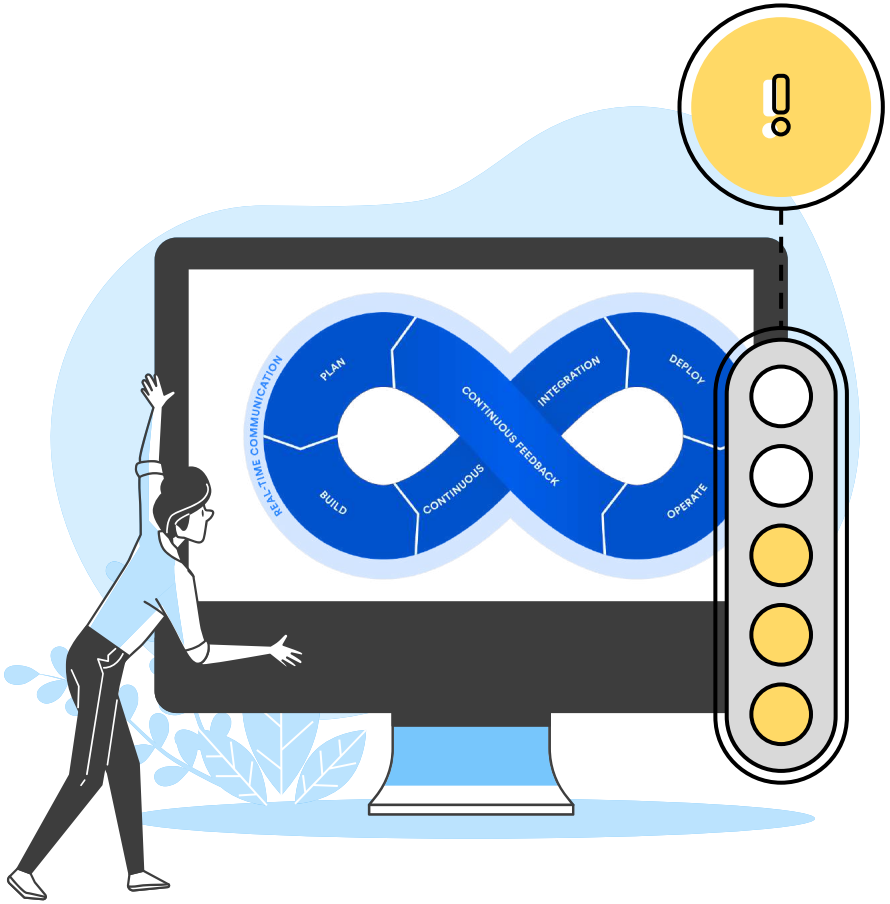


La capacidad de los desarrolladores de auto implantar código en producción, ver rápidamente los clientes satisfechos cuando sus recursos funcionan y corregir rápidamente cualquier problema sin tener que abrir un ticket en Operaciones.

3.5 Lanzamiento de Alto Riesgo

Proporcione implementaciones automatizadas en forma de autoservicio:

- Build
- Prueba
- Despliegue



Talleres Prácticos Sugeridos

3.5 Las Prácticas Técnicas Releases

Automatice y habilite las liberaciones de bajo riesgo (2 / 2)

En esta sección explicaremos :

1. Automatizar nuestro proceso de despliegue
2. Habilitar las implantaciones automatizadas de autoservicio
3. Integrar el despliegue de código en el proceso de despliegue
4. Desvincular las implantaciones de los lanzamientos
5. Patrones de despliegue basados en el entorno
6. El patrón de despliegue azul-verde
7. Cómo hacer frente a los cambios en las bases de datos
8. Los patrones de liberación del sistema inmune canario y del clúster
9. Patrones basados en la aplicación para permitir liberaciones más seguras
10. Implantar interruptores de funciones
11. Realizar lanzamientos oscuros
12. Estudio de la entrega continua y el despliegue continuo en la práctica

3.5 Lanzamiento de Bajo Riesgo

Arquitectura para liberaciones de bajo riesgo

En este capítulo, describiremos los pasos que podemos dar para invertir la espiral descendente, revisaremos los principales arquetipos arquitectónicos, examinaremos los atributos de las arquitecturas que permiten la productividad de los desarrolladores, la capacidad de prueba, la capacidad de despliegue y la seguridad, así como evaluaremos las estrategias que nos permiten migrar de forma segura desde cualquier arquitectura actual que tengamos a una que permita alcanzar mejor nuestros objetivos organizativos.

En esta sección explicaremos :

1. Una arquitectura que permite la productividad, la comprobabilidad y la seguridad
2. Arquetipos arquitectónicos: monolitos vs. Microservicios
3. Utilizar el patrón de aplicación estrangulador para evolucionar con seguridad nuestra arquitectura empresarial

3.5 Lanzamiento de Bajo Riesgo

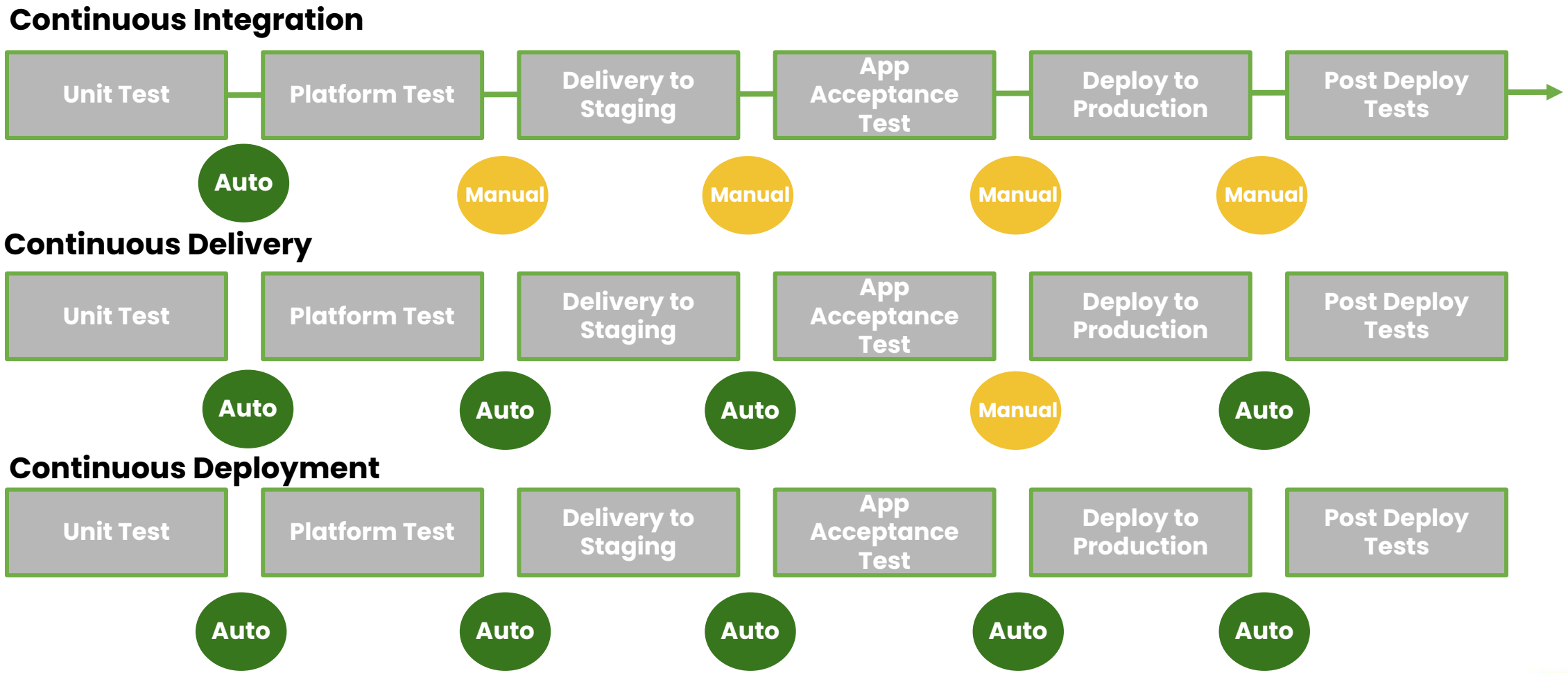
Automatice el proceso de Despliegue

Integre el código compilado en el pipeline de despliegue

Posterior al proceso de compilación de código automatizado, es posible incluirlo en el pipeline de despliegue.

- ✓ Garantizar que los paquetes sean adecuados para su despliegue en producción
- ✓ Mostrar rápidamente la preparación de los entornos de producción
- ✓ Proporcionar un método de autoservicio accionado por un botón
- ✓ Grabar automáticamente, qué comandos, en qué máquinas, cuándo, quién autorizó y cuál fue la salida
- ✓ Ejecutar una prueba de humo
- ✓ Proporcionar feedback rápido al implantador

3.5 Lanzamiento de Bajo Riesgo



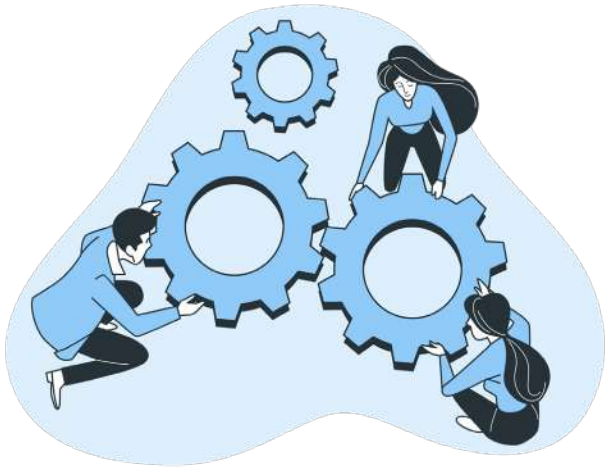
3.5 Lanzamiento de Bajo Riesgo

Desacoplar los despliegues de las liberaciones / releases

Necesitamos desacoplar nuestros despliegues de producción de nuestras liberaciones de aspectos o funcionalidad.

En la práctica, los términos despliegue y liberación se utilizan frecuentemente de forma intercambiable. Sin embargo, son dos acciones distintas que sirven a dos propósitos muy diferentes:

- Despliegue(Deployment)
- Liberación (release)



Los despliegues (deployment) *es la instalación de una versión específica del software en un entorno determinado.*

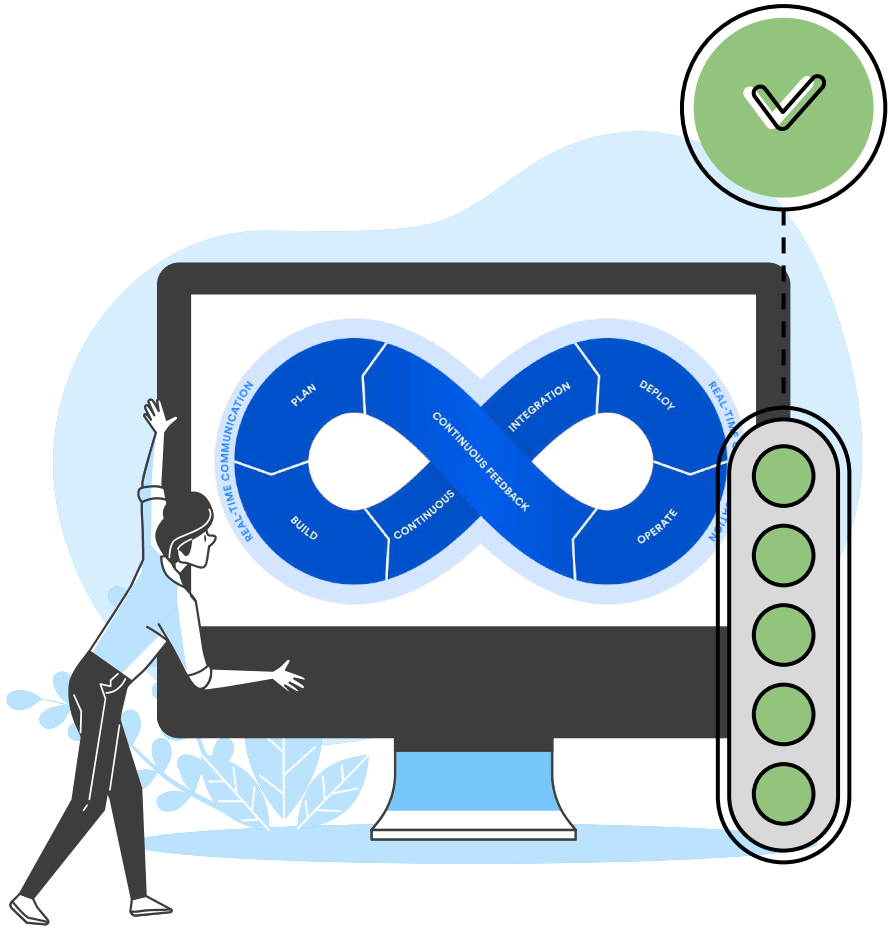
Liberación (release) *es cuando ofrecemos una funcionalidad (o conjunto) a todos nuestros clientes o a un segmento de clientes*

3.5 Lanzamiento de Bajo Riesgo

Automatice el proceso de Despliegue

Proporcione implementaciones automatizadas en forma de autoservicio:

- Build
- Prueba
- Despliegue
- Liberación (Operate)



3.5 Lanzamiento de Bajo Riesgo

Desacoplar los despliegues de las liberaciones / releases

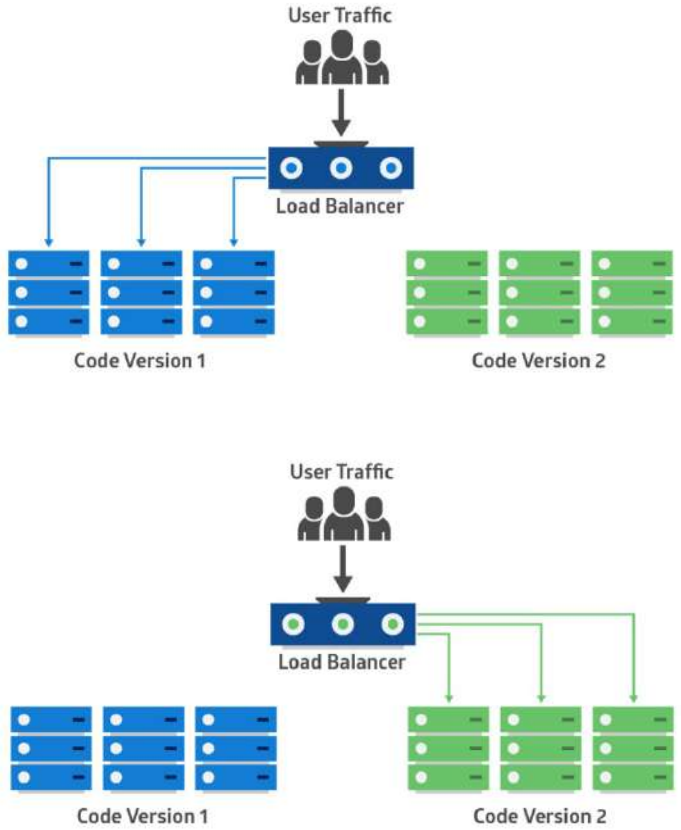
Hay dos grandes categorías de estándares de lanzamiento que podemos utilizar:

- **Estándares de versión basados en el entorno**: dos o más entornos en los que desplegamos, pero sólo un entorno está recibiendo tráfico de clientes activos (por ejemplo, configurando nuestros equilibradores de carga). El nuevo código se despliega en un entorno no activo y el lanzamiento se ejecuta moviendo el tráfico a ese entorno
- **Estándares de versión basados en aplicaciones**: es donde modificamos nuestra aplicación para que podamos liberar y exponer selectivamente la funcionalidad específica de la aplicación a través de pequeños cambios en la configuración. Por ejemplo, podemos desplegar nuevas funcionalidades que exponen progresivamente nuevas funcionalidades en producción

3.5 Lanzamiento de Bajo Riesgo

Desacoplar los despliegues de las liberaciones / releases

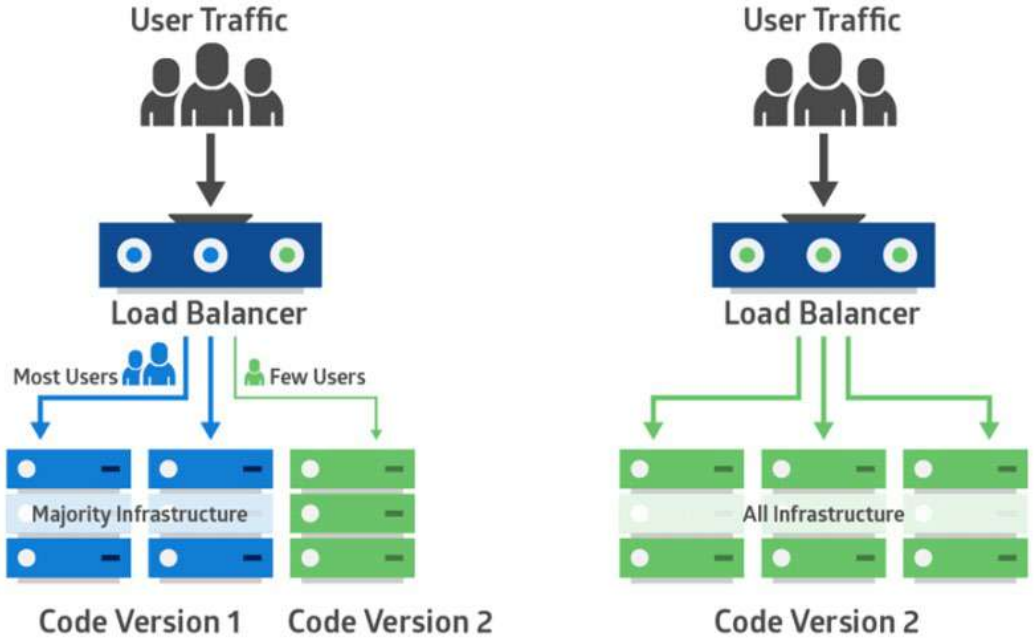
- **Estándares de versión basados en el entorno**: dos o más entornos en los que desplegamos, pero sólo un entorno está recibiendo tráfico de clientes activos (por ejemplo, configurando nuestros equilibradores de carga). El nuevo código se despliega en un entorno no activo y el lanzamiento se ejecuta moviendo el tráfico a ese entorno



Fuente: *https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a3*

3.5 Lanzamiento de Bajo Riesgo

- **Estándares de versión basados en aplicaciones**: es donde modificamos nuestra aplicación para que podamos liberar y exponer selectivamente la funcionalidad específica de la aplicación a través de pequeños cambios en la configuración. Por ejemplo, podemos desplegar nuevas funcionalidades que exponen progresivamente nuevas funcionalidades en producción



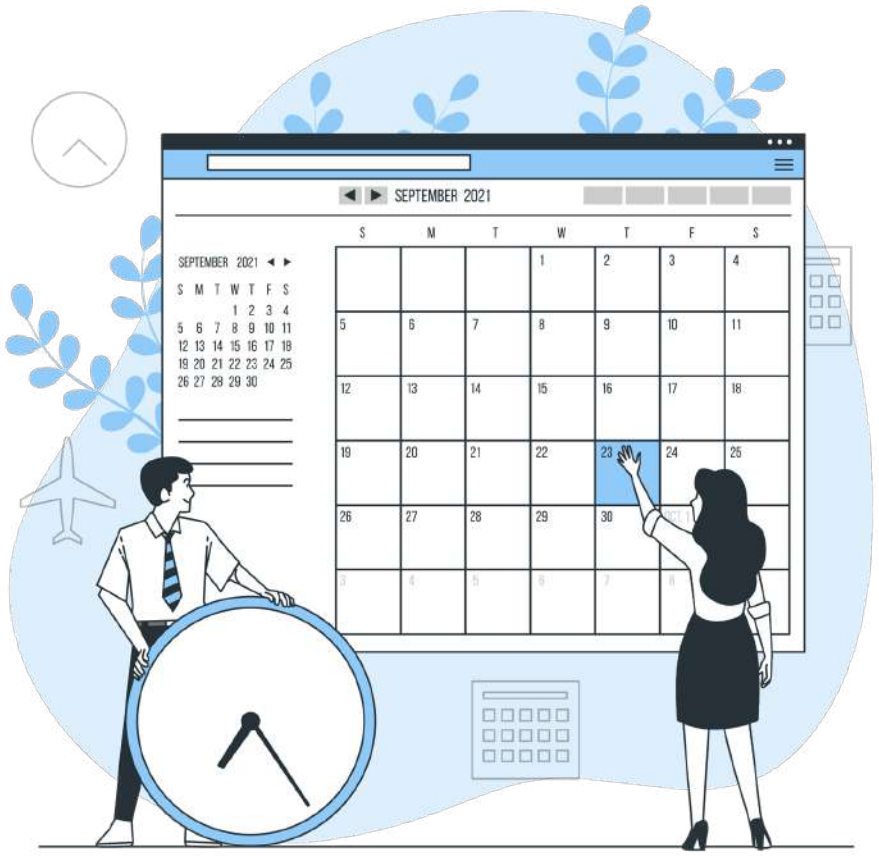
Fuente: https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a3

3.5 Lanzamiento de Bajo Riesgo

Estándares de versión basados en el entorno

Desacoplar despliegues de nuestros lanzamientos cambia drásticamente como trabajamos.

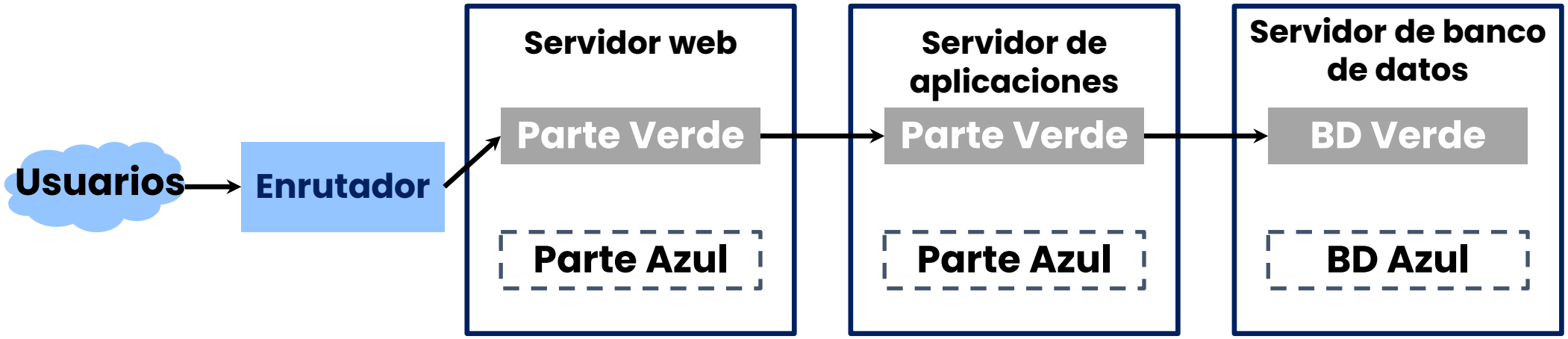
- Se evita despliegues en el medio de la noche o en los fines de semana
- Así, los despliegues son durante el horario de trabajo típico
- De esta manera, se disminuye significativamente el riesgo asociado a las liberaciones de producción y reducir el lead time de el despliegue



3.5 Lanzamiento de Bajo Riesgo

Estándar de liberación Azul-verde:

El más simple de los tres estándares se llama el despliegue azul-verde. En este estándar, hay dos entornos de producción: azul y verde. En cualquier momento, sólo uno de ellos está atendiendo al tráfico de clientes, el entorno verde está en vivo.

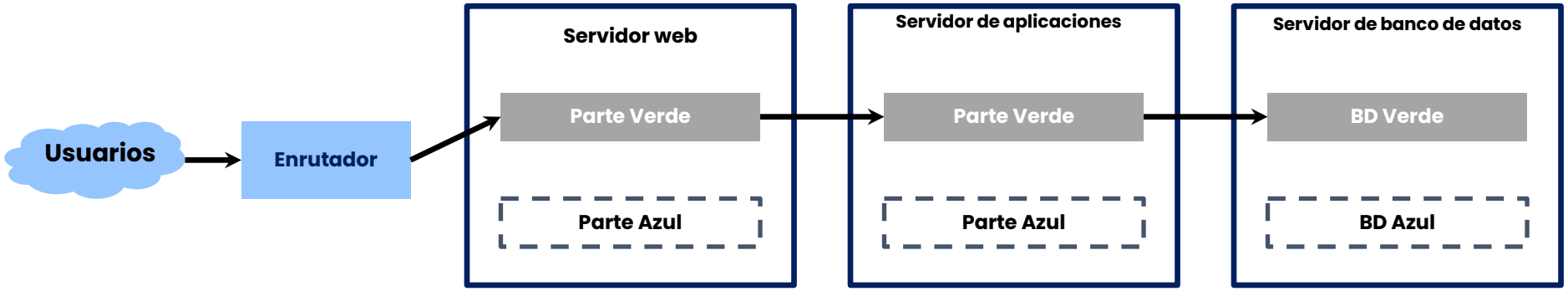


3.5 Lanzamiento de Bajo Riesgo

Lidiando con los cambios en la base de datos

Tener dos versiones de nuestra aplicación en producción crea problemas cuando dependen de una base de datos común. Hay dos enfoques generales para resolver este problema:

- Cree dos bases de datos (por ejemplo, una base de datos azul y otra verde)
- Desacoplar cambios de base de datos de cambios de aplicaciones



Este estándar también es comúnmente llamado estándar de expansión/contrato. No cambiamos (mutate) objetos de base de datos, como columnas o tablas. En vez de eso, primero hemos expandido, añadiendo nuevos objetos, y luego contratamos removiendo los antiguos.

3.5 Lanzamiento de Bajo Riesgo

Estándar de liberación Canario

Automatiza el proceso de lanzamiento de promoción para entornos más grandes y críticos, con confirmaciones planificadas.

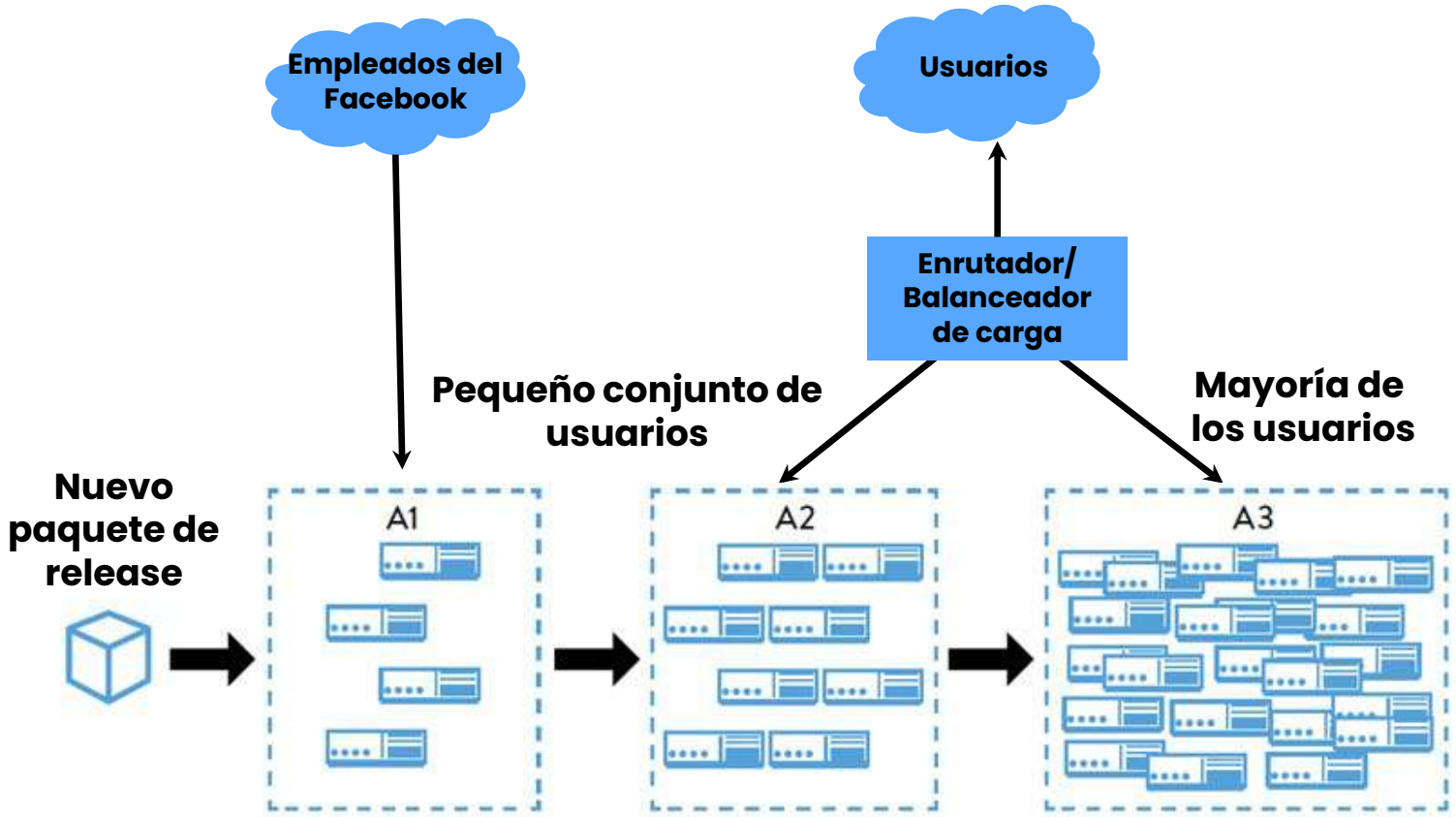
Se monitorea el rendimiento del software en cada entorno.

Cuando algo parece estar equivocado, retrocedemos; de lo contrario, se despliega en el siguiente entorno.

El término liberación de canario viene de la tradición de mineros de carbón, que llevaban canarios enjaulados a las minas para proporcionar detección precoz de niveles tóxicos de monóxido de carbono. Si hubiera mucho gas en la cueva, él mataría a los canarios antes de matar a los mineros, alertándolos para evacuar.

3.5 Lanzamiento de Bajo Riesgo

Ejemplo muestra los grupos de entornos que creó Facebook para soportar este estándar de release:



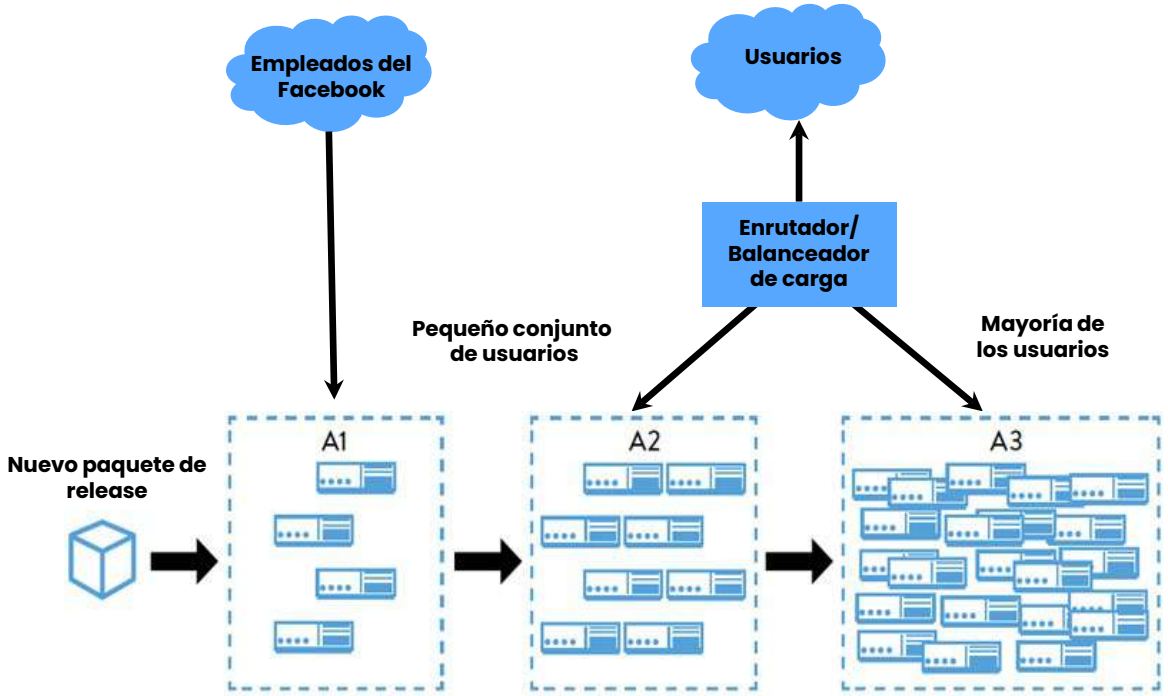
3.5 Lanzamiento de Bajo Riesgo

El sistema inmune de clúster

Expande el estándar de release canario conectando el sistema de monitoreo de producción con el proceso de release y automatizando la reversión del código.

Hay dos beneficios significativos:

1. Protegemos contra defectos que son difíciles de encontrar
2. Reducimos el tiempo necesario para detectar y responder al desempeño degradado



3.5 Lanzamiento de Bajo Riesgo

Aumenta la flexibilidad en la forma en que lanzamos nuevos recursos con seguridad para nuestros clientes, generalmente en una base por característica o funcionalidad.

Hay una necesidad de exigir mayor disciplina y participación del Desarrollo.

3.5 Lanzamiento de Bajo Riesgo

Despliega las funcionalidades selectivamente.

Mecanismo para habilitar y deshabilitar selectivamente recursos sin requerir un despliegue de código de producción. Las **alternancias de recursos** también nos permiten hacer lo siguiente:

- ✓ Retroceder fácilmente
- ✓ Degradar el rendimiento graciosamente
- ✓ Aumentar nuestra resiliencia a través de una arquitectura orientada a servicios

Proporciona el mecanismo para habilitar y deshabilitar selectivamente recursos sin requerir despliegues de código de producción.

3.5 Lanzamiento de Bajo Riesgo

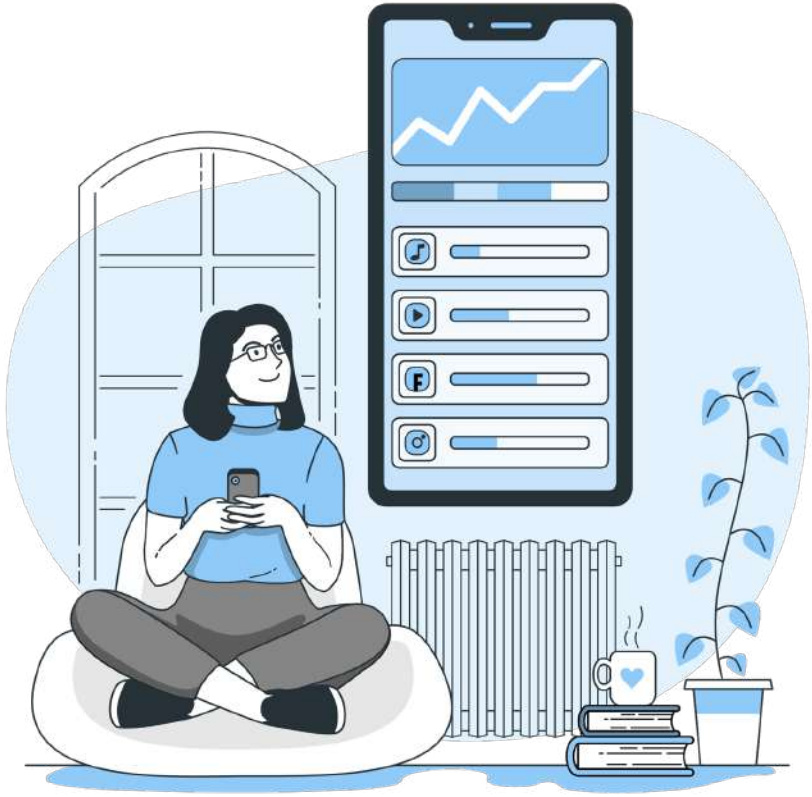
Desplegar Dark Releases/ Lanzamientos Ocultos

Las alternancias de recursos permiten desplegar recursos en la producción sin hacerlos accesibles a los usuarios, permitiendo una técnica conocida como lanzamiento oscuro.

Despliegues con toda funcionalidad en la producción:

- Realizamos pruebas de esta funcionalidad
- Todavía es invisible para los clientes

Para cambios grandes o arriesgados, muchas veces lo hacemos por semanas antes del lanzamiento en la producción, lo que nos permite probar con seguridad las cargas previstas de producción.



Talleres Prácticos Sugeridos

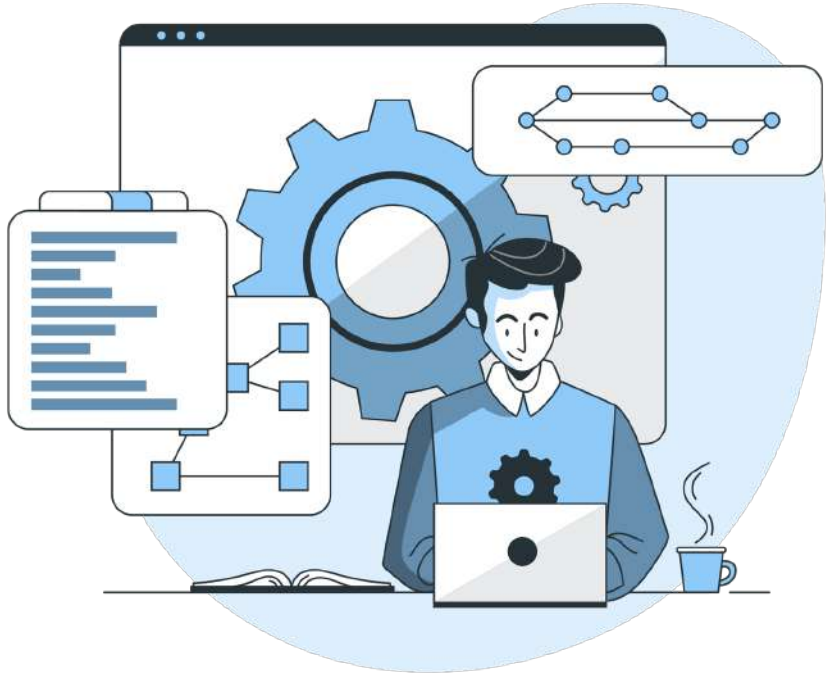
3.6 Arquitectura de Bajo Riesgo

Arquitectura evolucionista

El desafío es cómo seguir migrando de la arquitectura que tenemos para la arquitectura que necesitamos.

Migración:

- Buscar áreas de mayor retorno posible clasificando las páginas del sitio por los ingresos producidos
- Así, elegir las áreas de mayor ingreso



Técnica de estrangulamiento - en lugar de "extraer y reemplazar" servicios antiguos por arquitecturas que ya no soportan las metas organizacionales, se coloca la funcionalidad existente detrás de una API y se evita hacer más cambios.

3.6 Arquitectura de Bajo Riesgo

Arquitectura que permita productividad, testeabilidad y seguridad

Arquitectura débilmente acoplada:

- Interfaces bien definidas que refuerzan cómo los módulos se conectan
- Promueve la productividad y la seguridad

Permite que equipos pequeños, productivos y de dos pizzas puedan hacer pequeños cambios que puedan ser desplegados de manera segura e independiente.

Esta arquitectura orientada a servicios permite:

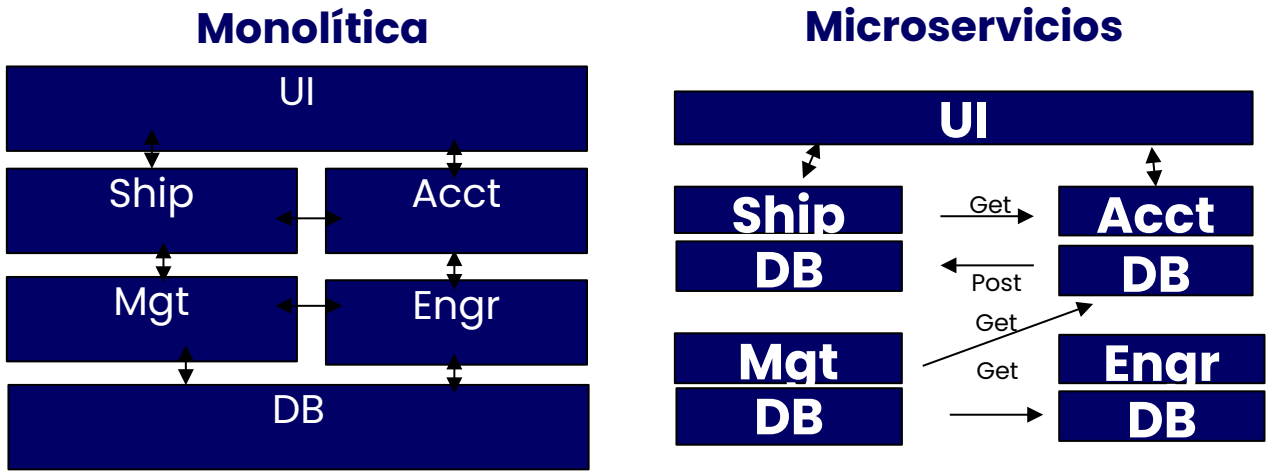
- Pequeños equipos trabajen en unidades más pequeñas y más sencillas de desarrollo
- Que cada equipo puede desplegar de forma independiente, rápida y segura

3.6 Arquitectura de Bajo Riesgo

Arquitectura Monolítica versus Microservicios

En algún momento de su historia, la mayoría de las organizaciones de DevOps fue perjudicada por arquitecturas monolíticas y fuertemente acopladas que, a pesar de ser extremadamente exitosas en ayudarlas a alcanzar el ajuste del producto/mercado, colocaron al grupo en riesgo de falla organizacional.

- Las arquitecturas monolíticas no son inherentemente malas - en realidad, por lo general, son la mejor opción para una organización al principio del ciclo de vida de un producto



3.6 Arquitectura de Bajo Riesgo

Utilice el patrón estrangulador para evolucionar la arquitectura con seguridad

Si hemos determinado que nuestra arquitectura actual está muy acoplada, podemos empezar a separar con seguridad partes de la funcionalidad de nuestra arquitectura existente.

Al hacer esto, habilitamos a los equipos que soportan la funcionalidad desacoplada a desarrollar, probar y desplegar independientemente su código en producción con autonomía y seguridad, además de reducir la entropía arquitectónica.

Creando aplicaciones estranguladoras, evitamos simplemente reproducir la funcionalidad existente en alguna nueva arquitectura o tecnología - generalmente, nuestros procesos de negocio son mucho más complejos de lo necesario debido a las idiosincrasias de los sistemas existentes, que acabamos reproduciendo.

Talleres Prácticos Sugeridos

Parte 4: La Segunda Forma – FEEDBACK

4.0 Las Prácticas Técnicas de la Retroalimentación

¿Por qué la Segunda Forma?

En el ámbito de la tecnología, nuestro trabajo se desarrolla casi por completo en sistemas complejos con un alto riesgo de consecuencias catastróficas. Como en la fabricación, a menudo descubrimos problemas sólo cuando se producen grandes fallos, como una interrupción masiva de la producción o una de producción o un fallo de seguridad que provoque el robo de datos de los clientes.

Hacemos que nuestro sistema de trabajo sea más seguro creando un flujo de información rápido, frecuente y de alta calidad de alta calidad a través de nuestro flujo de valor y nuestra organización, que incluye bucles de retroalimentación y alimentación. Esto nos permite:

- Detectar y remediar problemas mientras aún son pequeños, más baratos y más fáciles de arreglar
- Evitar los problemas antes de que causen una catástrofe, y
- Crear un aprendizaje organizativo que que integramos en el trabajo futuro
- Tratar a los fallos y accidentes, cómo oportunidades de aprendizaje, en lugar de una causa de castigo y culpa

4.0 Las Prácticas Técnicas de la Retroalimentación

Puntos a cubrir

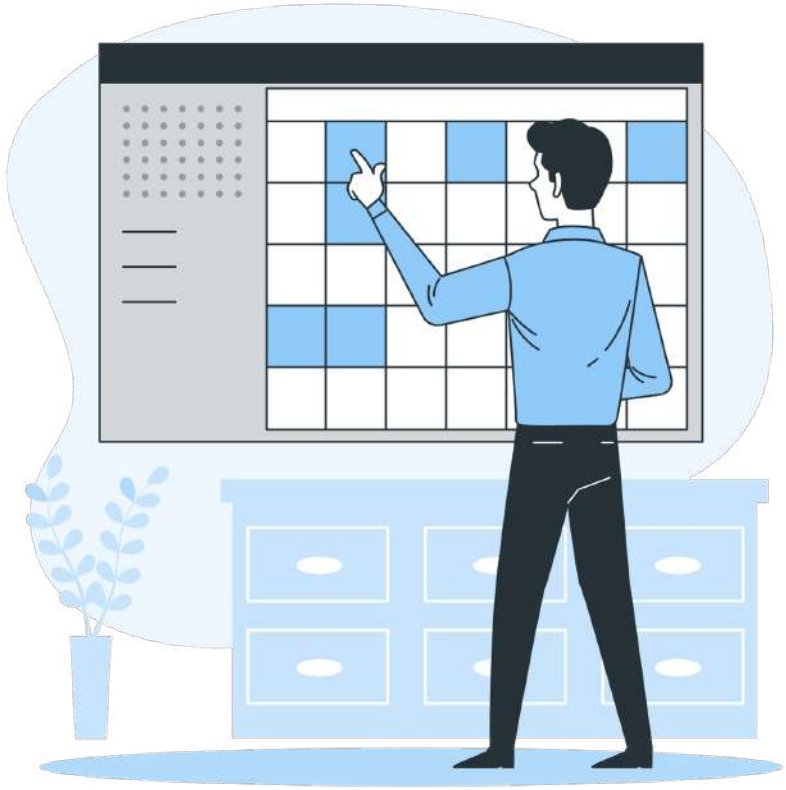
Mientras que la Primera Vía describe los principios que permiten el rápido flujo de traba de izquierda a derecha, la Segunda Vía describe los principios que permiten la de la derecha a la izquierda en todas las etapas del flujo de valor. Nuestro objetivo es crear un sistema de trabajo cada vez más seguro y resiliente.

En esta sección explicaremos:

1. Cómo crear telemetría para poder ver y resolver problemas
2. Utilizar nuestra telemetría para anticipar mejor los problemas y alcanzar los objetivos
3. Integrar la investigación y el feedback de los usuarios en el trabajo de los equipos de producto
4. Permitir la retroalimentación para que los departamentos de desarrollo y operaciones puedan realizar los despliegues de forma segura
5. Permitir la retroalimentación para aumentar la calidad de nuestro trabajo a través de revisiones por pares y la programación por pares

Agenda

- 4.1 Cómo crear telemetría para poder ver y resolver problemas.
- 4.2 Integrar la investigación y el feedback de los usuarios en el trabajo de los equipos de producto
- 4.3 Permitir la retroalimentación para que los departamentos de desarrollo y operaciones puedan realizar los despliegues de forma segura
- 4.4 Permitir la retroalimentación para aumentar la calidad de nuestro trabajo a través de revisiones por pares y la programación por pares



4.1 Telemetría: ¿Por qué?

Crear telemetría para permitir ver y resolver problemas

Un hecho de la vida en Operaciones, o cuando trabajamos con sistemas complejos, es que las cosas van a salir mal: pequeños cambios pueden llevar a resultados inesperados, incluyendo cortes y fallos globales que afectan a todos nuestros clientes. Esta es la realidad de la gestión de sistemas complejos: ninguna persona puede ver todo el sistema y entender cómo encajan todas las piezas.

Durante una interrupción al servicio, es posible que no se pueda determinar si el problema se produce debido a una falla:

- En nuestra aplicación (por ejemplo, defecto en el código)
- En nuestro entorno (por ejemplo, un problema de red, problema de configuración del servidor)
- Algo totalmente externo a nosotros (por ejemplo, un ataque masivo de denegación de servicio)

Ante esta situación, debemos usar un enfoque disciplinado para resolver problemas, utilizando la telemetría de la producción para entender los posibles factores que están contribuyendo al evento y así poder enfocarse en la solución de problemas, a diferencia de simplemente "reiniciar los servidores" a cada rato.

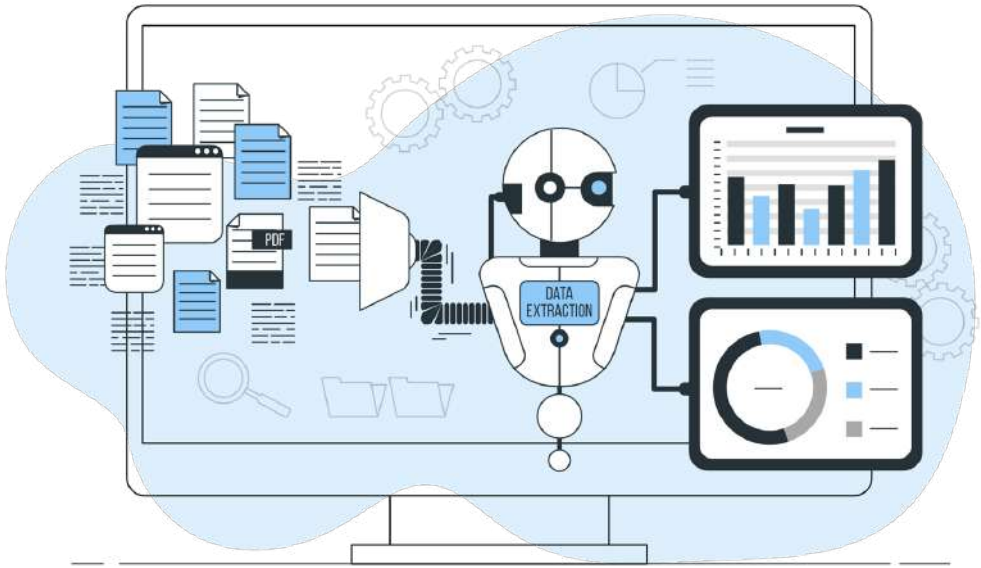
4.1 Telemetría: ¿Por qué?

La telemetría cómo habilitador en el diagnóstico

Para habilitar este comportamiento disciplinado de resolución de problemas, tenemos que diseñar nuestros sistemas para crear continuamente telemetría.

Telemetría se define como "un proceso de comunicación automatizado por el que se recogen mediciones y otros datos en puntos remotos y se transmiten a equipos receptores para su control".

Nuestro objetivo es crear telemetría dentro de nuestras aplicaciones y entornos, tanto en nuestra pre-producción como en los ambientes productivos, incluyendo nuestra cadena de despliegue.



Cultura de Causalidad: las organizaciones de mejor desempeño resultaron ser mucho más efectivas tanto en el diagnóstico, cómo en la corrección de los incidentes de servicio que sus pares.

4.1 Telemetría: ¿Por qué?

¿Por qué la telemetría?

En la Conferencia Velocity de 2012, McDonnell describió cuánto riesgo eso creaba:

"Estábamos cambiando algunas de nuestras infraestructuras más críticas, que, idealmente, los clientes nunca notarían. Sin embargo, ellos definitivamente notarán si nosotros estropeemos algo. Necesitábamos más métricas para darnos la confianza de que no estábamos realmente rompiendo las cosas mientras hacíamos estos grandes cambios, tanto para nuestros equipos de ingeniería y para miembros del equipo en áreas no técnicas, como marketing".

"Comenzamos a recoger toda la información de nuestro servidor en una herramienta de nombre Ganglia, mostrando toda la información en Graphite, una herramienta de código abierto en la que invertimos pesadamente. Al hacer esto, podríamos ver más rápidamente cualquier efecto colateral de el despliegue no intencional. Incluso empezamos a poner pantallas de televisión en toda la oficina para que todos pudieran ver el rendimiento de nuestros servicios".

4.1 Telemetría: ¿Por qué?

Mejora la capacidad de resolver incidentes

Las organizaciones de alta performance resuelven incidentes de producción 168 veces más rápido que sus pares, con un promedio de alto rendimiento que tiene un MTTR medido en minutos, mientras que la mediana de bajo rendimiento tuvo un MTTR medido en días. Las dos principales prácticas técnicas que permitieron el MTTR rápido fueron:

- El uso del control de versiones por Operaciones
- La utilización de telemetría y monitoreo proactivo en el entorno de producción

El objetivo es garantizar que siempre tengamos telemetría suficiente para poder confirmar que nuestros servicios están funcionando correctamente en todos nuestros ambientes.

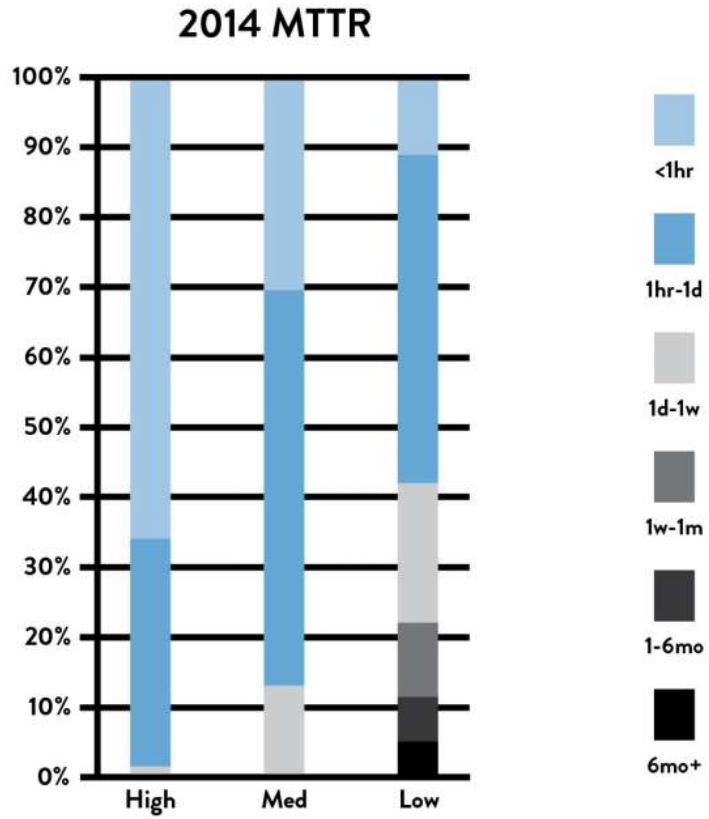


Figure 25: *Incident resolution time for high, medium, and low performers (Source: Puppet Labs, 2014 State of DevOps Report.)*

Fuente: DevOps Handbook

4.1 Telemetría: ¿Por qué?

Resumen

El objetivo es asegurar que siempre tengamos suficiente telemetría para que podamos confirmar que nuestros servicios están funcionando correctamente en producción.

Y cuando se produzcan problemas, hacer que sea posible determinar rápidamente lo que va mal y tomar decisiones informadas sobre la mejor manera de solucionarlo, idealmente mucho antes de que los clientes se vean afectados.

Además, la telemetría es lo que nos permite reunir nuestra mejor comprensión de la realidad y detectar cuando nuestra comprensión de la realidad es incorrecta.

4.1 Telemetría: Arq. de Monitoreo Moderna

Infraestructura de Telemetría Centralizada

Durante décadas hemos acabado con silos de información, donde Desarrollo sólo crea eventos de registro que son útiles para los desarrolladores, y Operaciones sólo supervisa si los entornos están activos o inactivos.

El resultado es que cuando ocurren eventos inoportunos, nadie puede determinar por qué todo el sistema no está funcionando como se diseñó o qué componente específico está fallando, impidiendo nuestra capacidad de devolver el sistema a su estado de funcionamiento.

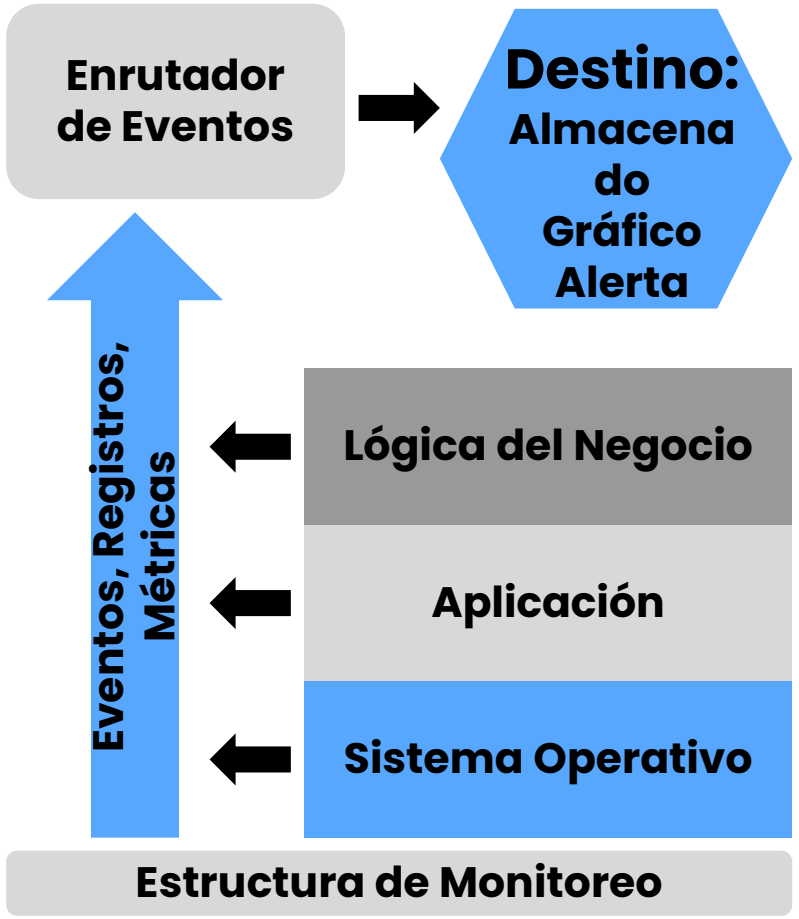
Debemos diseñar y desarrollar nuestras aplicaciones y entornos para que generen telemetría suficiente, permitiéndonos entender cómo nuestro sistema se está comportando como un todo.

4.1 Telemetría: Arq. de Monitoreo Moderna

Infraestructura de Telemetría Centralizada y Moderna

Cuando todos los niveles de nuestro ecosistema de aplicaciones tienen monitoreo y registro, habilitamos otros recursos importantes, como gráficos y visualización de nuestras métricas, detección de anomalías, alerta proactiva y escalonamiento, etc. Para ello se sugiere:

- Recolección de datos en la capa de lógica de negocios, aplicaciones y entornos: para crear telemetría en forma de eventos, logs y métricas
- Un enrutador de eventos responsable de almacenar nuestros eventos y métricas: este recurso permite la visualización, la tendencia, la alerta, la detección de anomalías, etc



4.1 Telemetría: Arq. de Monitoreo Moderna

Arquitectura de monitoreo moderna

- Al centralizar los registros:
  - Podemos convertirlos en métricas que podemos contabilizar en el enrutador de eventos
- Al transformar los logs en métricas:
  - Es posible realizar operaciones estadísticas, como usar la detección de anomalías para encontrar valores discrepantes y variaciones más rápidamente

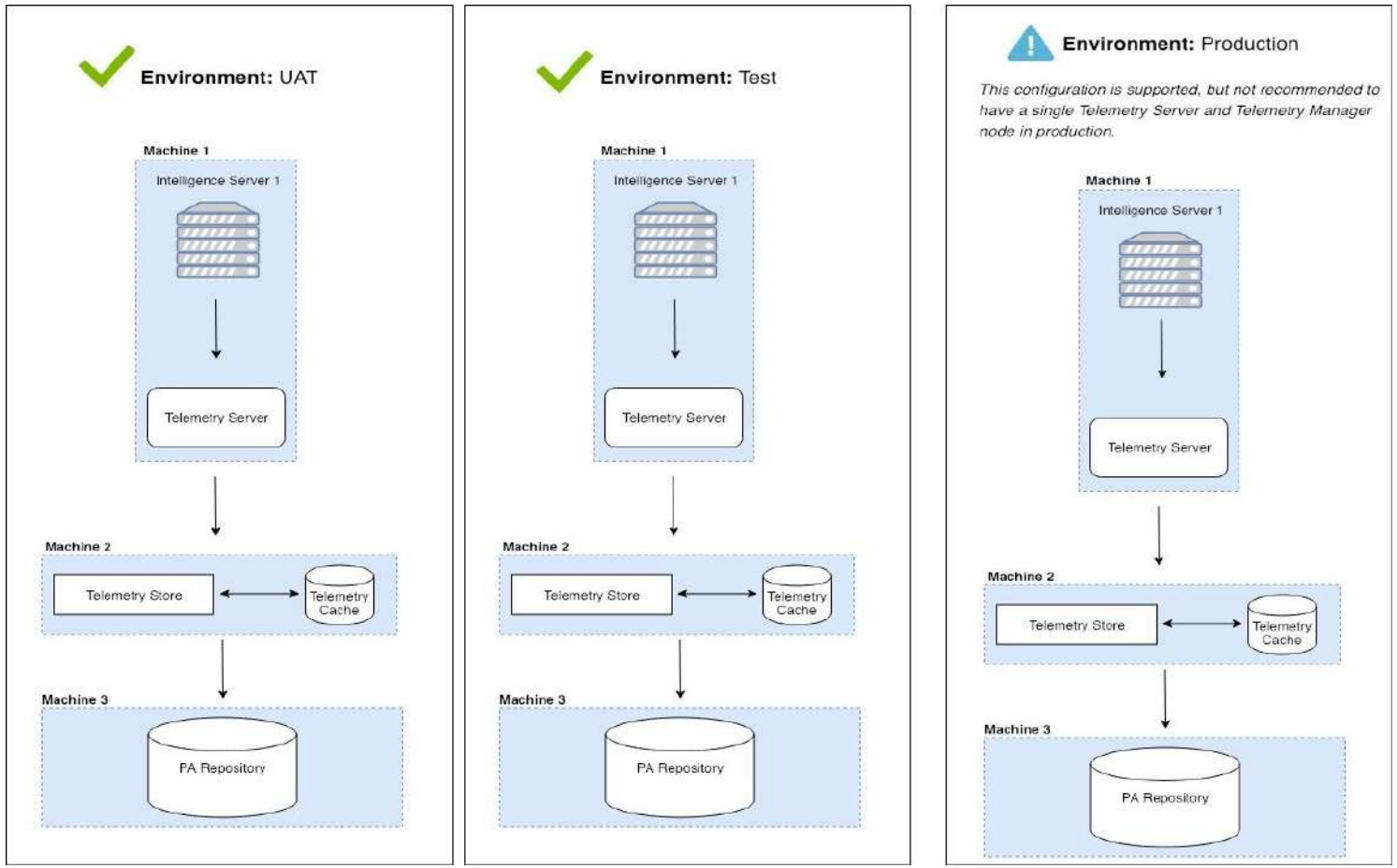
4.1 Telemetría: Arq. de Monitoreo Moderna

Adicionalmente, podemos recoger telemetría en el pipeline de despliegue en eventos importantes, ej.: las pruebas automatizadas que pasan o fallan o en los despliegues que se están realizando en cualquiera de los entornos.

También puedo recoger telemetría de la duración o el tiempo que se está tomando el ejecutar nuestros Builds o corridas de pruebas de regresión.

Al hacer esto, podemos detectar condiciones que podrían indicar problemas, como por ejemplo si la prueba de rendimiento o nuestra compilación tarda el doble de lo normal, lo que nos permite encontrar y corregir los errores antes de que pasen a producción.

4.1 Telemetría: Arq. de Monitoreo Moderna



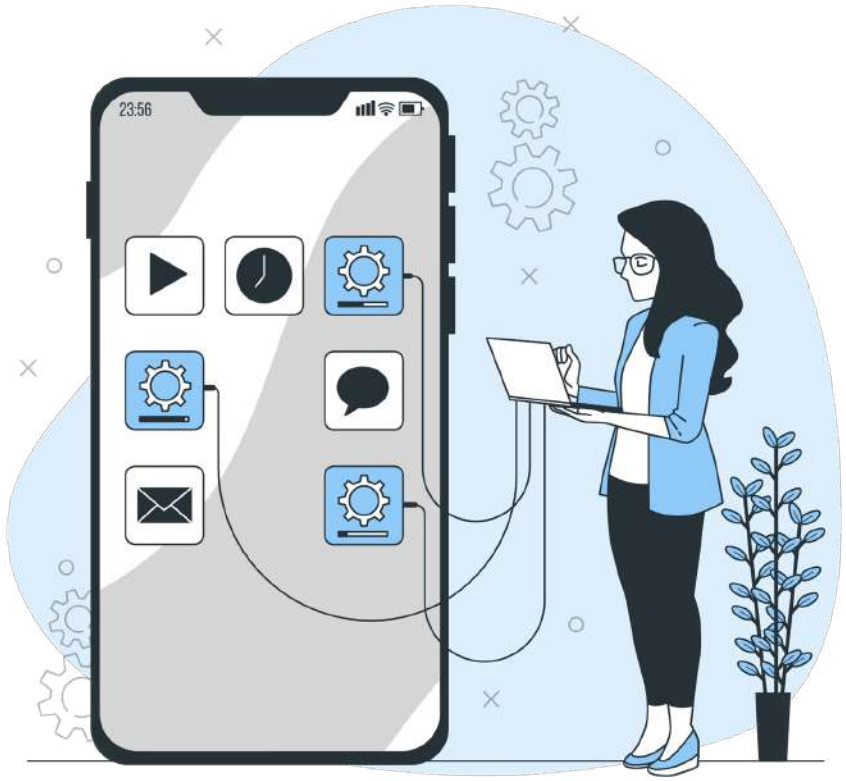
https://www2.microstrategy.com/producthelp/Current/PlatformAnalytics/en-us/Content/pa_architecture_examples.htm

4.1 Telemetría: Logging de Aplicaciones

Crear Telemetría De Registro De Aplicaciones Que Ayuda A La Producción

Todos los miembros del pipeline utilizarán la telemetría de varias maneras:

- El equipo *DEV* pueden crear temporalmente más telemetría en su aplicación para diagnosticar mejor los **problemas en su local**
- El equipo *OPS* puede usar la telemetría para diagnosticar un **problema de producción**
- *Infosec* y los auditores pueden revisar la telemetría para confirmar la eficacia de un control necesario, rastrear los **resultados de negocios, el uso de recursos o las tasas de conversión**

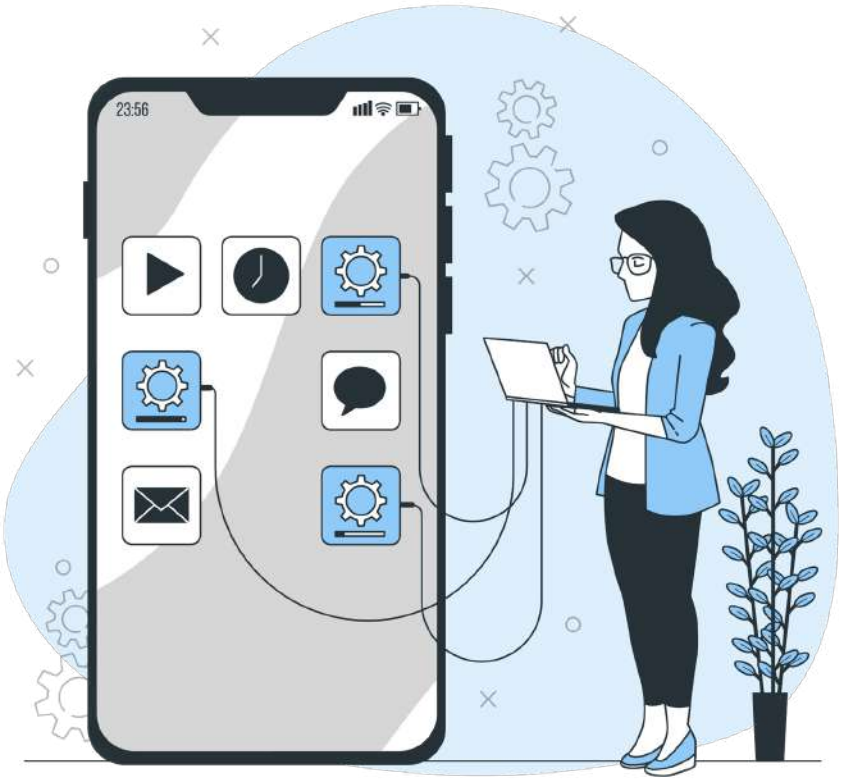


4.1 Telemetría: Logging de Aplicaciones

Crear registros de telemetría

Para soportar estos varios modelos de uso, tenemos diferentes niveles de registro, algunos de los cuales también pueden accionar alertas, como los siguientes:

- **Nivel de DEBUG**
- **Nivel INFO**
- **Nivel de WARN (alerta)**
- **Nivel de ERROR (excepción)**
- **Nivel FATAL**



4.1 Telemetría: Logging de Aplicaciones

Crear registros de telemetría

Todos los eventos potencialmente significativos de la aplicación deben generar entradas de registro, incluidas las que figuran en esta lista elaborada por Anton A. Chuvakin, vicepresidente de investigación del grupo GTP Security and Risk Management de Gartner:

- Decisiones de autenticación/autorización (incluido el cierre de sesión)
- Acceso al sistema y a los datos
- Cambios en el sistema y las aplicaciones (especialmente los cambios de privilegios)
- Cambios en los datos, como añadir, editar o eliminar datos
- Entradas no válidas (posibles inyecciones maliciosas, amenazas, etc.)
- Recursos (RAM, disco, CPU, ancho de banda o cualquier otro recurso que tenga límites)
- Salud y disponibilidad del servicio
- Arranques y paradas
- Fallos y errores
- Disparos de circuit breakers
- Retrasos / Delays
- Éxito/fallo de las copias de seguridad

4.1 Telemetría: Logging de Aplicaciones

Logging para mejorar el análisis de los incidentes

La telemetría nos permite utilizar el método científico para formular hipótesis sobre la causa de un problema concreto y lo que se necesita para resolverlo. Ejemplos de preguntas que podemos responder durante la resolución del problema incluyen:

- ¿Qué pruebas tenemos de nuestra monitorización de que un problema se está produciendo realmente?
- ¿Cuáles son los eventos y cambios relevantes en nuestras aplicaciones y entornos que podrían haber contribuido al problema?
- ¿Qué hipótesis podemos formular para confirmar la relación entre las causas y los efectos propuestos?
- ¿Cómo podemos demostrar cuáles de estas hipótesis son correctas y solucionar el problema con éxito?

El valor de la resolución de problemas basada en hechos no sólo radica en un mejor MTTR (y mejores resultados para el cliente), sino también en el refuerzo de la percepción de una relación de beneficio mutuo entre Desarrollo y Operaciones.

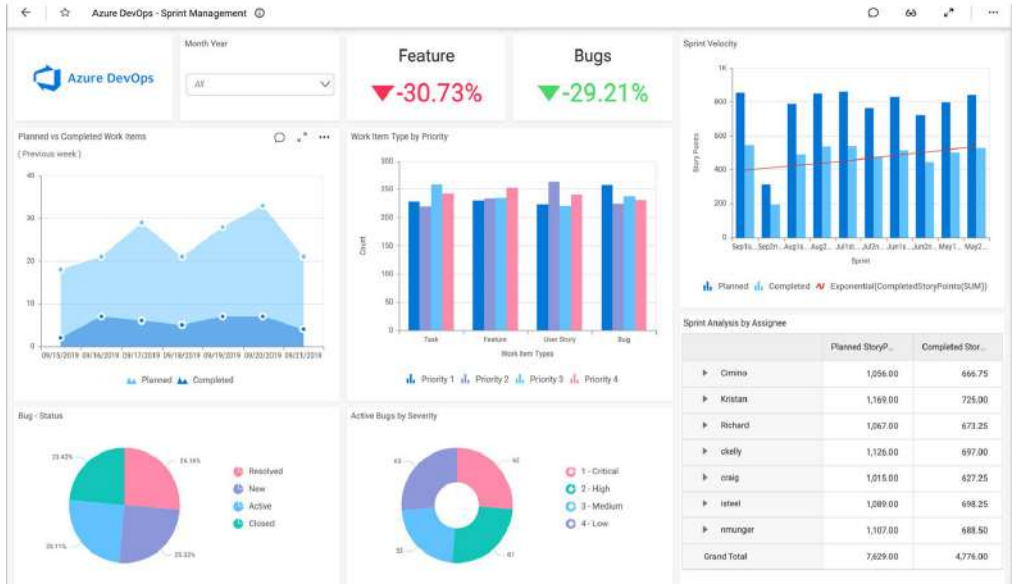
4.1 Telemetría: Radiadores de Información

Crear Autoservicios para Acceso a Radiadores De Información y Telemetría

Queremos que nuestra telemetría de producción sea muy visible, lo que significa ponerla en áreas centrales donde trabajen Desarrollo y Operaciones, permitiendo así que todos los que estén interesados vean cómo están funcionando nuestros servicios (Desarrollo, Operaciones, Gestión de Productos e Infosec. entre otros)

Se promueve la responsabilidad entre los miembros del equipo, demostrando activamente valores:

- El equipo no tiene nada que esconder de sus visitantes (clientes, partes interesadas, etc.)
- El equipo no tiene nada que esconder de sí mismo: reconoce y enfrenta problemas



https://www.boldbi.com/integrations/azure-devops

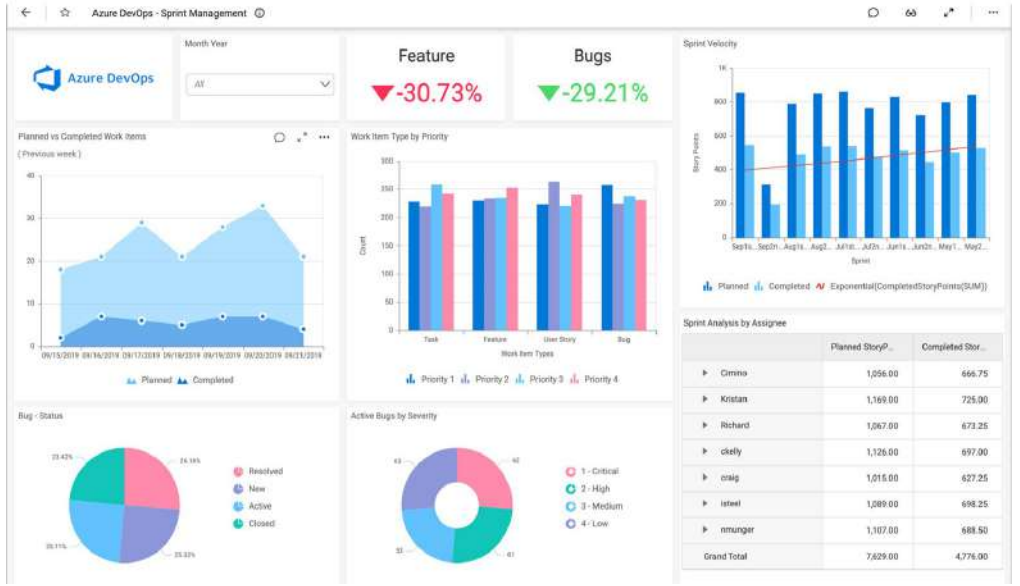
4.1 Telemetría: Brechas en la Telemetría

Encontrar y rellenar las lagunas en la Telemetría

Para conseguirlo es necesario crear suficiente telemetría en todas las gaps de las aplicaciones y para todos nuestros entornos, así como para los pipelines de despliegue que los soportan. Necesitamos métricas de los siguientes niveles:

- Nivel de negocio
- Nivel de aplicación
- Nivel de infraestructura
- Nivel de software del cliente
- Nivel de pipelines de despliegue

Al tener cobertura de telemetría en todas estas áreas, podremos ver la salud de todo aquello en lo que se basa nuestro servicio, utilizando datos y hechos en lugar de rumores, acusaciones, culpas, etc.



https://www.boldbi.com/integrations/azure-devops

4.1 Telemetría: Brechas en la Telemetría

Encontrar y rellenar las lagunas en la Telemetría

- **Nivel de negocio:** Los ejemplos incluyen el número de transacciones de ventas, los ingresos de las transacciones de ventas, las inscripciones de usuarios, la tasa de abandono, los resultados de las pruebas A/B, etc
- **Nivel de aplicación:** Los ejemplos incluyen los tiempos de transacción, los tiempos de respuesta de los usuarios, fallos de la aplicación, etc
- **Nivel de infraestructura** (por ejemplo, base de datos, sistema operativo, red almacenamiento): Los ejemplos incluyen el tráfico del servidor web, la carga de la CPU, el uso del disco, etc
- **Nivel de software del cliente** (por ejemplo, JavaScript en el navegador del cliente, aplicación móvil): Los ejemplos incluyen errores y fallos de la aplicación, tiempos de transacción medidos por el usuario, etc.
- **Nivel de Pipeline de despliegue**: Los ejemplos incluyen el estado de la cadena de construcción (por ejemplo, rojo o verde para nuestros diversos conjuntos de pruebas automatizados), tiempos de despliegue de cambios de despliegue, frecuencias de despliegue, promociones de entornos de prueba y estado del entorno

4.1 Telemetría: Análisis Predictivos Telemetría

Analizar la telemetría para anticiparse mejor a los problemas y alcanzar los objetivos

Como vimos en la sección anterior necesitamos suficiente telemetría de producción en nuestras aplicaciones e infraestructura para ver y resolver los problemas a medida que se producen. Es importante que creemos herramientas que nos permitan descubrir variaciones y señales de fallo cada vez más débiles ocultas en nuestra telemetría de producción para poder evitar fallos catastróficos. Algunas técnicas que podemos usar son:

- Utilizar las medias y las desviaciones estándar para detectar posibles problemas
- Instrumentar y alertar sobre resultados no deseados
- Utilizar técnicas de detección de anomalías

4.1 Telemetría: Análisis Predictivos Telemetría

Estas técnicas estadísticas pueden utilizarse para analizar nuestra telemetría de producción, de modo que podamos encontrar y solucionar los problemas antes que ocurran, a menudo cuando todavía son pequeños y mucho antes de que causen resultados catastróficos. Esto nos permite encontrar señales de fallo cada vez más pequeñas sobre las que podemos actuar, creando un sistema de trabajo cada vez más seguro.

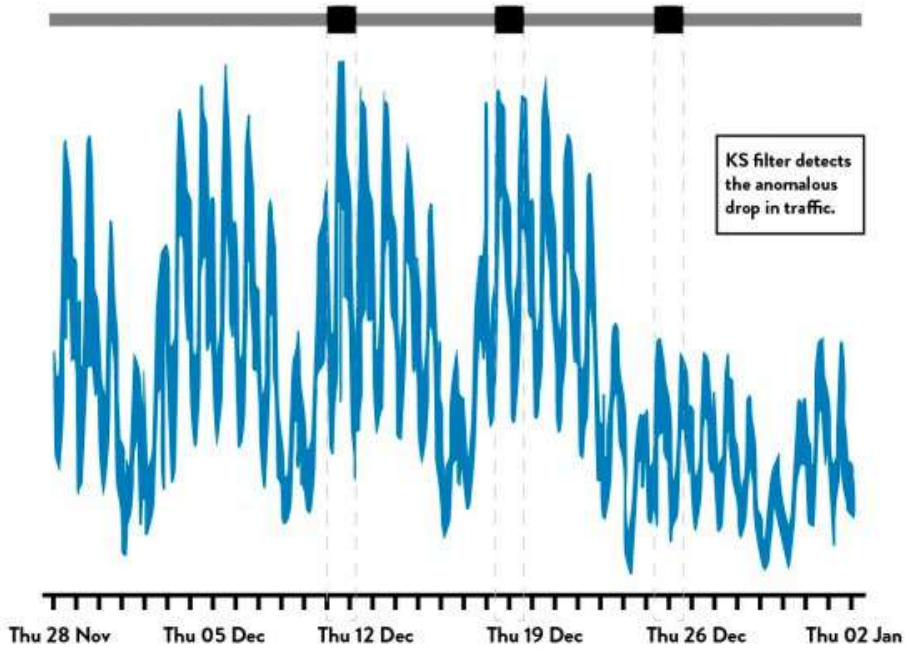


Figure 36: *Transaction volume: using Kolmogorov-Smirnov test to alert on*

Fuente: DevOps Handbook

4.2 Feedback

Permitir la retroalimentación para que el desarrollo y las operaciones puedan desplegar el código con seguridad

En esta sección hablaremos de los mecanismos de retroalimentación que nos permiten mejorar nuestro servicio en cada etapa de nuestro trabajo diario, ya sea desplegando cambios en producción, arreglando el código cuando las cosas van mal y se llama a los ingenieros, haciendo que los desarrolladores sigan su trabajo aguas abajo, creando requisitos no funcionales que ayuden a los equipos de desarrollo a escribir un código más listo para la producción, o incluso devolviendo servicios problemáticos para que sean autogestionados por Desarrollo.

4.2 Feedback

Ampliar los ciclos de feedback

Al crear estos bucles de retroalimentación, hacemos que los despliegues de producción sean más seguros, aumentamos la preparación para la producción del código creado por Desarrollo y ayudamos a crear una mejor relación de trabajo entre Desarrollo y Operaciones al reforzar los objetivos, las responsabilidades y la empatía compartidos.

Algunas acciones para mejorar el feedback:

- Usar la telemetría para hacer el proceso de despliegue más seguro
- Rotación del Standby entre Operaciones y Desarrollo
- Hacer que los desarrolladores acompañan los procesos aguas abajo (downstream)
- Hacer que los desarrolladores autogestionen al principio su servicio en producción
  - Launch Guidance
  - Launch Readiness Review
  - Hand Off Readiness Review

4.2 Feedback

Aumente la telemetría para liberaciones más seguras

Monitorear activamente las métricas asociadas a las funcionalidades durante el despliegue, garantiza que no violamos nuestro servicio inadvertidamente o, peor, que violamos otro servicio.

Si el cambio se rompe o perjudica cualquier funcionalidad, trabajamos rápidamente para restaurar el servicio, trayendo a quien más necesite para diagnosticar y corregir el problema.

Se puede optar por:

- **Desactivar los recursos rotos** con la "Feature Toggles"
- **Corregir**
- **Dar Marcha atrás**

4.2 Feedback

Comparta deberes entre Dev y Ops (Stand-bys, Pager Rotation, Prod Issue Resolution, etc)

En cualquier servicio complejo todavía tendremos problemas inesperados, como incidentes e interrupciones que ocurren en momentos inoportunos y que se presentan de manera consistente (todas las noches a las 2 de la mañana). Estos, si no se corrigen, pueden causar problemas recurrentes con daño a las Operaciones e impacto a los usuarios.

Para evitar que esto ocurra, **todos los participantes** del flujo de valor comparten las responsabilidades de manejar los incidentes operativos. De este modo, el departamento de operaciones no se enfrenta, aislado y solo, a los problemas de producción relacionados con el código, sino que todo el mundo ayuda a encontrar el equilibrio adecuado entre la corrección de los defectos de producción y el desarrollo de nuevas funcionalidades, independientemente del lugar en el que nos encontremos en el flujo de valor.

4.2 Feedback

Dev Team acompaña aguas abajo su desarrollo

Una de las técnicas más poderosas en interacción y diseño de experiencia del usuario (UX) es la **investigación contextual**, que es cuando el equipo del producto observa a un cliente utilizar la aplicación en su entorno natural, generalmente trabajando en su escritorio.

Nuestro objetivo es utilizar esa misma técnica para observar cómo nuestro trabajo afecta a nuestros clientes internos. Los desarrolladores deben seguir su trabajo en sentido descendente, de modo que puedan ver cómo los centros de trabajo descendentes deben interactuar con su producto para llevarlo a producción.

Al hacer esto, creamos una retroalimentación sobre los aspectos no funcionales de nuestro código -todos los elementos que no están relacionados con la función de cara al cliente, e identificamos formas de mejorar la capacidad de despliegue, gestión y funcionamiento, etc.

4.2 Feedback

Dev teams auto gestionando sus servicios en producción al inicio

Cuando los **desarrolladores** han desplegando y están ya **ejecutando su código** en entornos de **producción** diariamente pueden surgir **errores graves**.

Aun equipos experimentados de Operaciones pueden experimentar lanzamientos de producción desastrosos porque es la primera vez que realmente vemos cómo nuestro código se comporta durante una liberación y bajo condiciones reales de producción.

Una contramedida potencial es tener **grupos de desarrollo auto gestionando sus servicios en producción** antes de ser elegibles para ser entregados a un grupo de operaciones administrado centralizado.

Al hacer que el **equipo de desarrollo** gestione inicialmente sus propias aplicaciones y servicios, el proceso de transición de nuevos servicios a la producción se hace mucho más **fácil y previsible**.

4.2 Feedback

Ops puede retornar un módulo a desarrollo nuevamente

Para servicios **ya en producción**, necesitamos un mecanismo diferente para garantizar que Ops nunca quede atrapada en un servicio no soportable en producción.

Podemos crear un **mecanismo de devolución (handback)**, cuando un servicio de producción se vuelve muy frágil, Ops tiene la capacidad de devolver la responsabilidad del soporte a la producción **de vuelta al Dev**.

Cuando un servicio retorna a un estado **administrado por el desarrollador**, la función de **Operaciones** pasa del soporte a la **producción para la consulta**, ayudando al equipo a preparar el servicio para la producción.

4.2 Feedback

Guia de Lanzamiento (Launch Guidance)

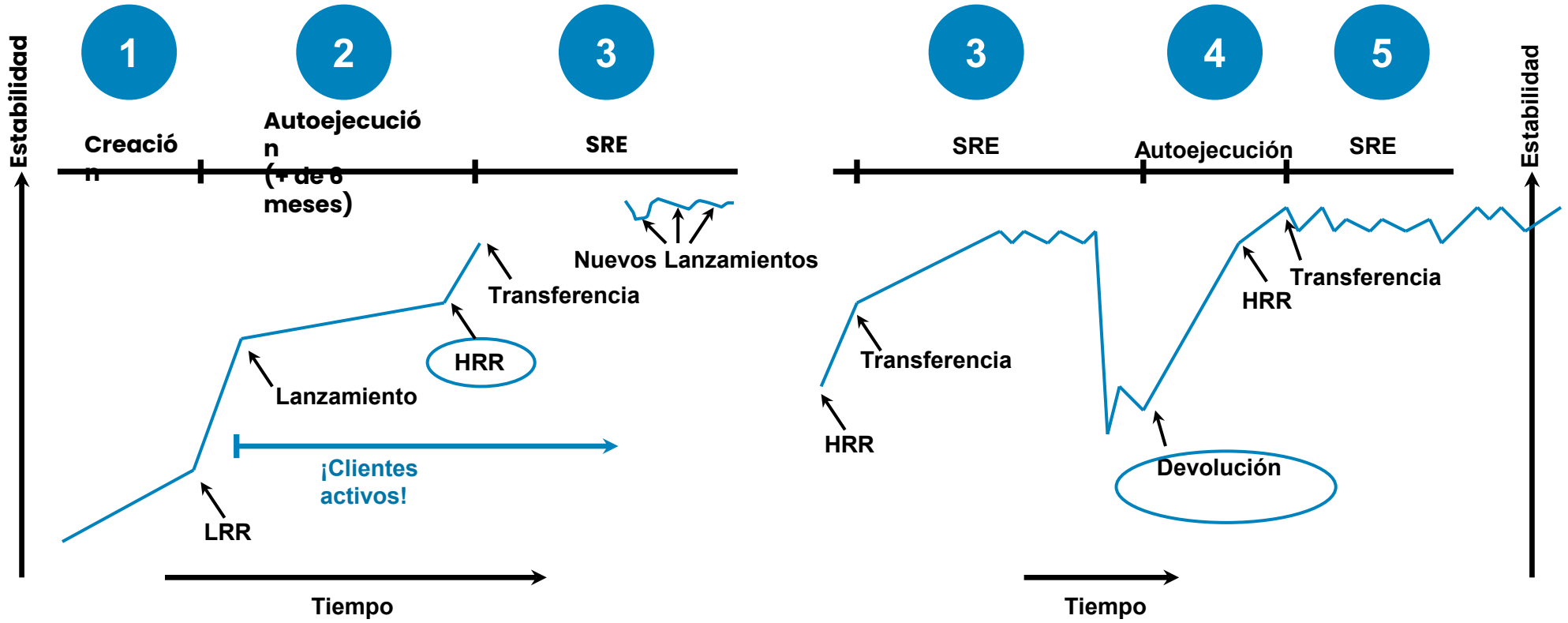
Al crear una guía de lanzamiento, ayudamos a garantizar que cada equipo de producto se beneficie de la experiencia acumulada y colectiva de toda la organización, especialmente de Operaciones. Las orientaciones y los requisitos de lanzamiento incluirán probablemente lo siguiente:

- **Recuento y gravedad de los defectos:** ¿La aplicación funciona realmente como se ha diseñado?
- **Tipo/frecuencia de las alertas de buscapersonas:** ¿Genera la aplicación un número insoportable de alertas en producción?
- **Cobertura de la supervisión:** ¿La cobertura de la supervisión es suficiente para restablecer el servicio cuando las cosas van mal?
- **Arquitectura del sistema:** ¿Está el servicio lo suficientemente desacoplado como para soportar un alto índice de cambios y despliegues en producción?
- **Proceso de despliegue: ¿**Existe un proceso predecible, determinista y suficientemente automatizado para desplegar el código en producción?
- **Higiene de producción: ¿**Existe evidencia de suficientes buenos hábitos de producción que permitan que el soporte de producción sea gestionado por cualquier otra persona?

4.2 Feedback

Superficialmente, estos requisitos pueden parecer similares a las listas de control de producción tradicionales que hemos utilizado en el pasado. Sin embargo, las diferencias clave son que requerimos una supervisión eficaz, que los despliegues sean fiables y deterministas, y una arquitectura que soporte despliegues rápidos y frecuentes.

Launch Readiness Review y Hand-Off Readiness Review



SRE: Ingeniero de Confiabilidad de Sitio
LRR: Revisión de Preparación de Lanzamiento
HRR: Revisión de Preparación para la Entrega

Fuente: DevOps Handbook

4.2 Feedback

Launch Readiness Review y Hand-Off Readiness Review

Google ha creado dos conjuntos de comprobaciones de seguridad para dos etapas críticas del lanzamiento de nuevos servicios, denominadas **Launch Readiness Review** y **Hand-Off Readiness Review** (LRR y HRR, respectivamente).

La **LRR** debe realizarse y aprobarse antes de que cualquier nuevo servicio de Google se ponga a disposición del público y reciba tráfico de producción en directo, mientras que la **HRR** se realiza cuando el servicio pasa a un estado gestionado por operaciones, normalmente meses después de la LRR.

4.2 Feedback

Las listas de comprobación de la LRR y la HRR son similares, pero la HRR es mucho más estricta y tiene normas de aceptación más estrictas, mientras que la LRR es autoinformada por los equipos de producto.

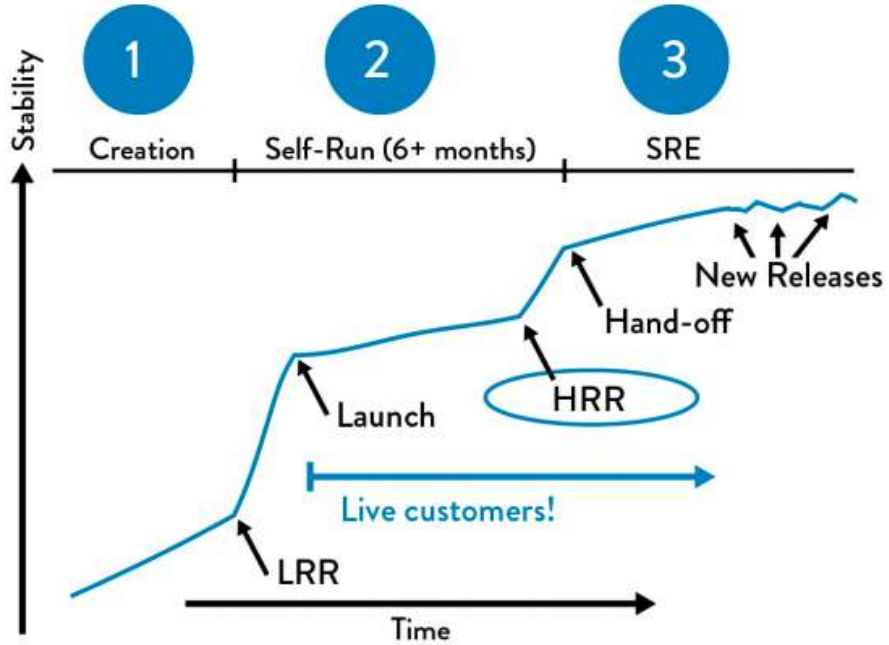


Figure 39: *The "Launch readiness review and hand-offs readiness review" at Google (Source: "SRE@Google: Thousands of DevOps Since 2004," YouTube video, 45:57, posted by USENIX, January 12, 2012, https://www.youtube.com/watch?v=iluTnhdTzK0.)*

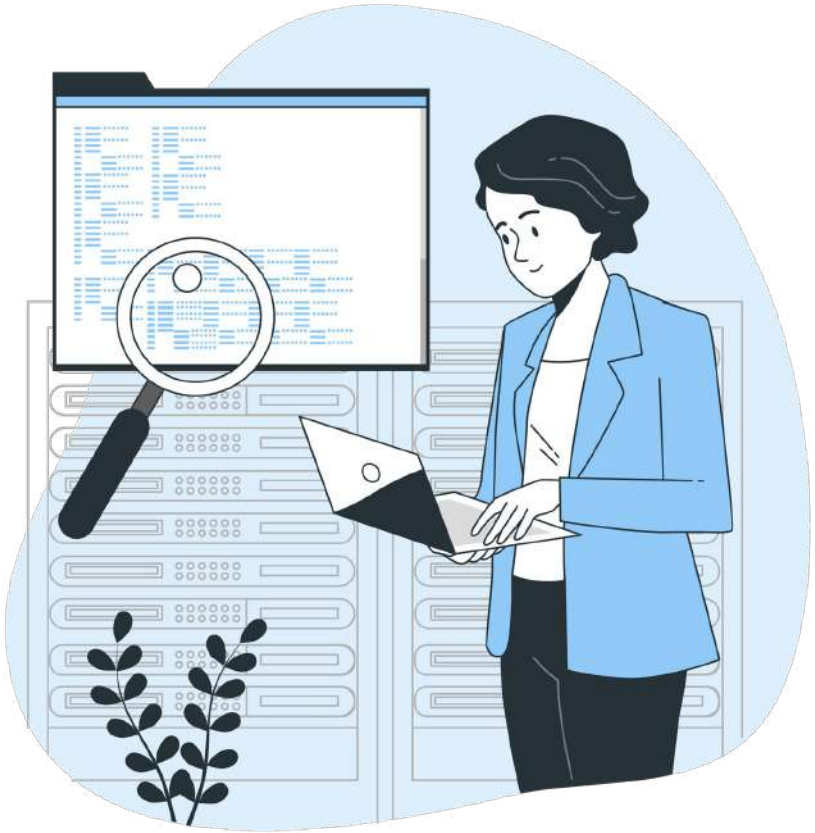
Fuente: DevOps Handbook

4.3 Desarrollo Basado en Hipótesis

Integrar el desarrollo basado en hipótesis y las pruebas A / B en nuestro trabajo diario

Con demasiada frecuencia, en los proyectos de software, los desarrolladores trabajan en las funciones durante meses o años, abarcando varias versiones, sin confirmar nunca si se están cumpliendo los resultados empresariales deseados, como por ejemplo si una función concreta está logrando los resultados deseados o incluso si se está utilizando.

"La forma más ineficiente de probar un modelo de negocio o una idea de producto es construir el producto completo para ver si la demanda prevista realmente existe" - Jez Humble.



4.3 Desarrollo Basado en Hipótesis

Antes de construir una funcionalidad, debemos preguntarnos:

- ¿Debemos construirlo? y
- ¿Por qué?

Luego, deberíamos realizar los experimentos más baratos y rápidos posibles para validar a través de la investigación del usuario si la función deseada realmente logrará los resultados deseados. Podemos utilizar técnicas cómo:

- Desarrollo impulsado por hipótesis
- Los embudos de adquisición de clientes
- Las pruebas A / B

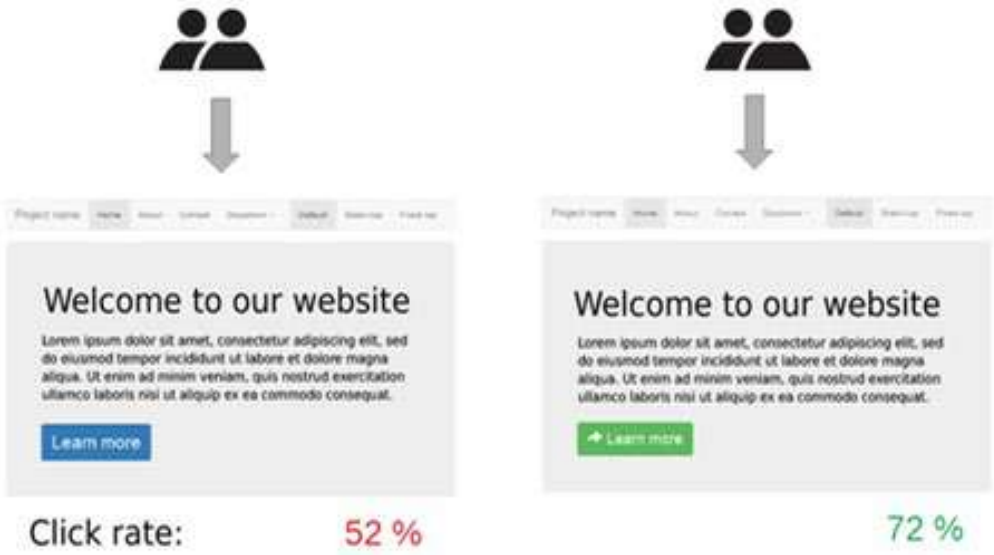
*"La manera más **ineficiente** de probar un modelo de negocio o una idea de producto es construir el **producto completo** para ver si la demanda prevista realmente existe".*

4.3 Desarrollo Basado en Hipótesis – Construcción de Funcionalidades

Integración de Prueba A/B en la validación de funcionalidades

Una técnica de búsqueda de usuarios es la definición del pipeline de adquisición de clientes y la realización de pruebas A / B.

La técnica A/B más comúnmente usada en la moderna práctica de UX implica un sitio en el que los visitantes se seleccionan aleatoriamente para exhibir una de las dos versiones de una página, un control (el "A") o un tratamiento (el "B").



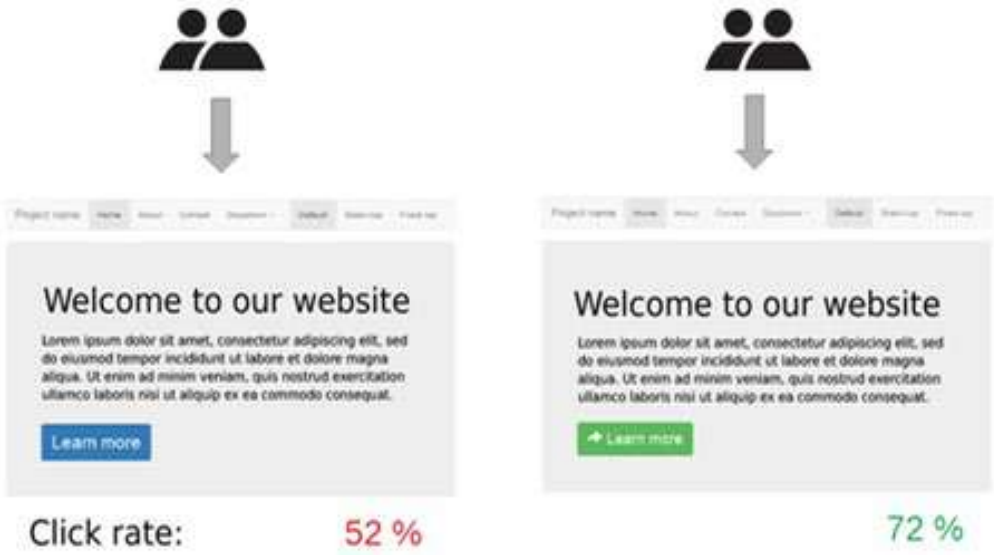
4.3 Desarrollo Basado en Hipótesis – Liberaciones

Integración de Prueba A/B en la liberación

Las pruebas A/B rápidas e iterativas son posibles gracias a la capacidad de realizar despliegues de producción de forma rápida y fácil bajo demanda.

- Esto requiere una **telemetría** de producción útil en todos los niveles del conjunto de aplicaciones

Al entrar en nuestro recurso o funcionalidad, se puede controlar qué porcentaje de los usuarios ve la versión de tratamiento de un experimento.



4.3 Desarrollo Basado en Hipótesis – Planificación

Integración de Prueba A/B en la planificación de funcionalidades

Una vez que tenemos la infraestructura para apoyar el lanzamiento y las pruebas de características A/B, debemos hacer que los Product Owners piensen en cada característica como una hipótesis y utilicen nuestros despliegues de producción como experimentos con usuarios reales para probar o refutar esa hipótesis.

La construcción de experimentos debe diseñarse en el contexto del general de adquisición de clientes. Barry O'Reilly, coautor de “Lean Enterprise: How High Performance Organizations Innovate at Scale” , describió cómo podemos enmarcar las hipótesis en el desarrollo de características.

4.3 Desarrollo Basado en Hipótesis – Planificación

Podemos enmarcar las hipótesis en el desarrollo de características de la siguiente forma:

- Creemos que aumentar el tamaño de las imágenes del hotel en la página de reservas dará lugar a una mejora del compromiso y la conversión de los clientes
- Tendremos confianza para proceder cuando veamos un aumento del 5% en clientes que revisan las imágenes del hotel y que luego proceden a reservar en cuarenta y ocho horas

Adoptar un enfoque experimental para el desarrollo de productos requiere no sólo dividir el trabajo en pequeñas unidades (historias o requisitos), sino también validar si cada unidad de trabajo genera los resultados esperados. Si no lo hace, modificamos nuestra hoja de ruta de trabajo con caminos alternativos que realmente alcancemos los resultados esperados.

4.4 Revisión de Pares

Crear procesos de revisión y coordinación para aumentar la calidad de nuestro trabajo actual

Nuestro objetivo es permitir que Desarrollo y Operaciones trabajen colaborativamente para reducir el riesgo que representa introducir cambios en la producción antes de que se realicen los despliegues.

Tradicionalmente, cuando revisamos los cambios para la implementación, tendemos a depender en gran medida de las revisiones, inspecciones y aprobaciones justo antes de la implementación. Con frecuencia, esas aprobaciones son otorgadas por equipos externos que a menudo están demasiado alejados del trabajo para tomar decisiones informadas sobre si un cambio es arriesgado o no, y el tiempo requerido para obtener todas las aprobaciones necesarias también alarga nuestro cambio.

En esta sección vamos a compartir unas prácticas técnicas más eficientes para reducir el riesgo de los despliegues en produccion.

4.4 Revisión y Coordinación

Crear procesos de revisión y coordinación para aumentar la calidad del trabajo

El proceso de **revisión por pares** en GitHub es un ejemplo impresionante de cómo la inspección puede aumentar la calidad, hacer que los despliegues sean más seguras y hacerlo integrándose al flujo de trabajo diario de todos.

Ellos fueron pioneros en el proceso llamado ***pull request***, una de las formas más populares de revisión por pares que abarca Dev y Ops.

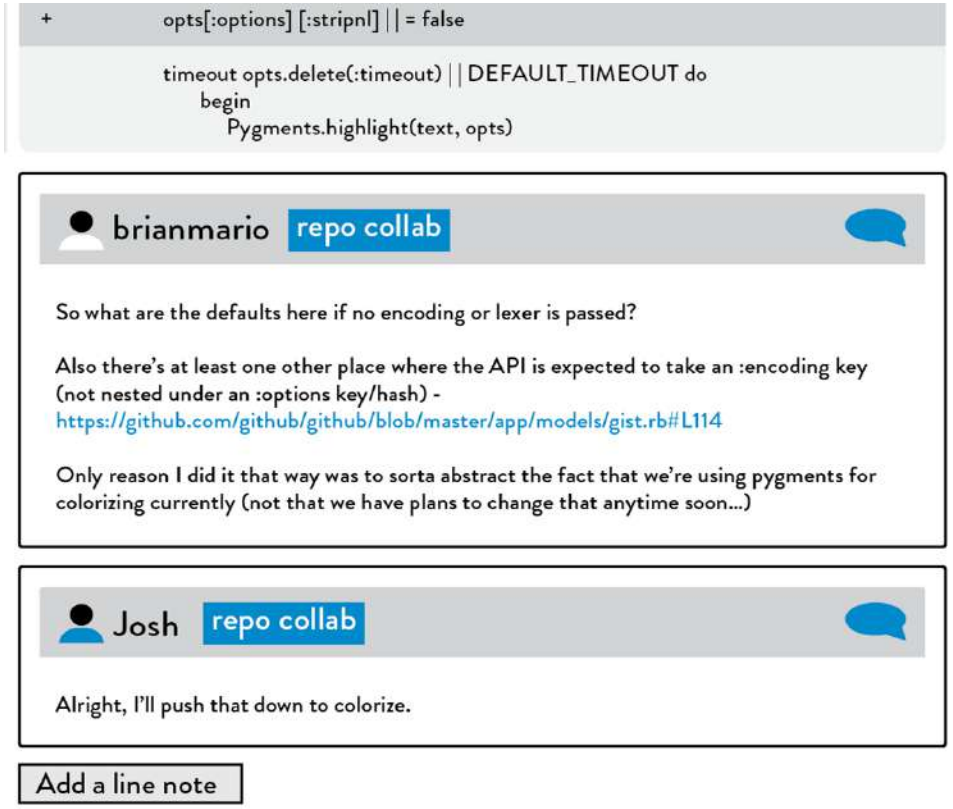
El objetivo ahora es permitir que el Desarrollo y las Operaciones reduzcan el riesgo de cambios en la producción antes de que se realicen.

4.4 Revisión y Coordinación

Crear procesos de revisión y coordinación

El Flujo de GitHub está compuesto de cinco etapas:

1) Para trabajar en algo nuevo, el desarrollador **crea una branch** local desde Master (por ejemplo, "new-auth2-scopes")
2) El desarrollador **efectúa sus cambios en ese branch localmente**, enviando regularmente su trabajo al mismo branch nombrado en el servidor
3) Cuando necesiten feedback o ayuda, o cuando crean que el branch está listo para ser “merged” crean un pull request
4) Cuando se hayan resuelto los comentarios al código y se obtengan las aprobaciones necesarias del recurso, el desarrollador podrá mezclarlo al maestro
5) Cuando los cambios de código se mezclan y se envían a master, el desarrollador las despliega en producción



4.4 Revisión y Coordinación

Pull Request

Un pull request de mala calidad es aquel que no tiene contexto suficiente para el lector, con poca o ninguna documentación de lo que el cambio pretende hacer.

Por ejemplo, un pull request que simplemente contiene el siguiente texto: "Corregir el problema 3616 y 3841".

Descripción recomendada en un pull request:

- Debe haber suficientes detalles sobre por qué se está haciendo el cambio
- Como el cambio fue hecho
- Cualquier riesgo identificado
- Contramedidas resultantes

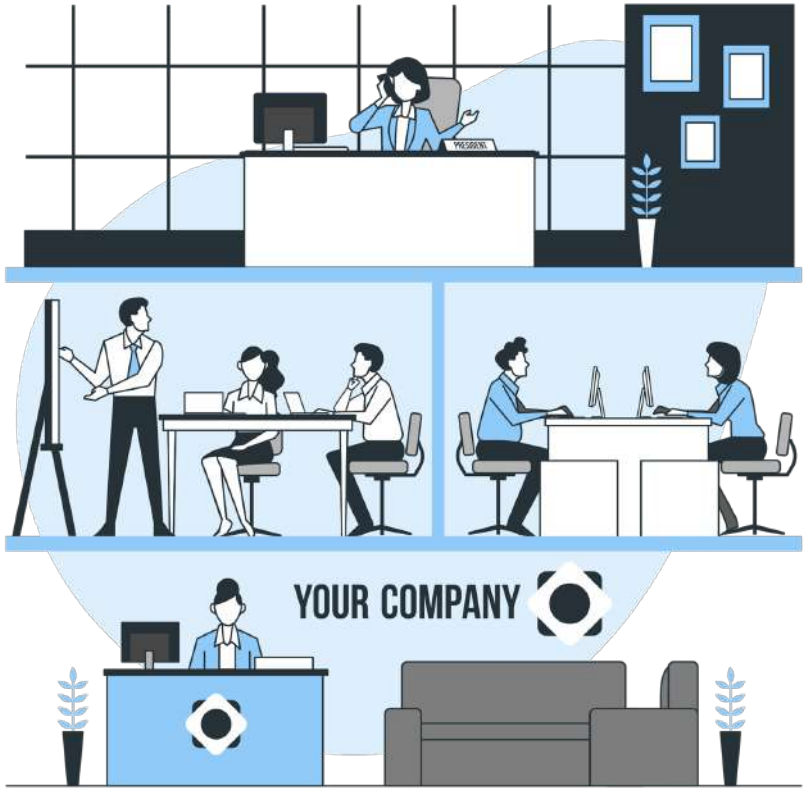
Información mejor de la solicitud tirada:

- Riesgos adicionales señalados
- Ideas sobre las mejores formas de desplegar el cambio deseado
- Ideas sobre cómo mejor atenuar el riesgo

4.4 Revisión y Coordinación

Peligros potenciales del “control excesivo”

Los mecanismo para la gestión de cambio (Change Management) tradicionales 00*pueden llevar a resultados negativos no intencionales, cómo lo son agregar retrasos adicionales a la entrega y una reducción la inmediatez del feedback necesario, y por lo tanto de su impacto en el proceso de despliegue.



“Las personas más cercanas a un problema generalmente saben más sobre él” – STP.

4.4 Revisión y Coordinación

Algunos de los controles que muchas veces implementamos cuando ocurren fallas en el proceso de control de cambios:

- Agregar más preguntas al formulario de solicitud de cambio
- Exigir más autorizaciones, como más un nivel de aprobación de gestión (por ejemplo, el vicepresidente, el CIO)
- Exigir más tiempo de preparación para aprobaciones de cambios, para que se evalúen adecuadamente

Estos controles generalmente agregan más fricción al proceso de despliegue, multiplicando el número de etapas y aprobaciones, y aumentando el tamaño de los lotes y los plazos de implementación.

"Las personas más cercanas a un problema generalmente saben más sobre él" – STP.

4.4 Revisión y Coordinación

Una de las creencias fundamentales del Sistema de Producción Toyota es que "las personas más cercanas a un problema suelen ser las que más saben sobre él". Esto se acentúa a medida que el trabajo que se realiza y el sistema en el que se produce el trabajo se vuelven más complejo y dinámico, como es típico en los flujos de valor de DevOps.

Como se ha demostrado repetidas veces, cuanto mayor es la distancia entre la persona que realiza el trabajo (es decir, el implementador del cambio) y la persona que decide sobre hacerlo (es decir, el autorizador de cambio), peor el resultado.

"Las personas más cercanas a un problema generalmente saben más sobre él" – STP.

4.4 Revisión y Coordinación

Coordinación y programación del cambio

Siempre que tengamos varios grupos trabajando en sistemas que comparten dependencias es probable que haya que coordinar nuestros cambios para garantizar que no interfieran entre sí (por ejemplo, la organización, la agrupación y la secuenciación de los cambios).

En informática y diseño de sistemas, un **sistema poco acoplado** es aquel en el que cada uno de sus componentes hace uso, o tiene poca o ninguna interacción con las funciones de otros componentes separados. Las subáreas incluyen el acoplamiento de clases, interfaces, datos y servicios. Sistemas **altamente acoplados** son los sistemas que están altamente interconectados.

Para mitigar riesgos, podemos utilizar salas de chat (tipo Slack) para anunciar cambios y localizar de manera proactiva posibles colisiones.

4.4 Revisión y Coordinación

En general:

- Cuanto más desacoplada sea nuestra arquitectura, menos tendremos comunicación y coordinación con otros equipos de componentes. Si arquitectura está realmente orientada a los servicios, los equipos pueden realizar cambios con un alto grado de autonomía, y es poco probable que los cambios locales provoquen interrupciones globales
- Sin embargo, incluso en una arquitectura poco acoplada, cuando muchos equipos realizan cientos de despliegues independientes al día, puede existir el riesgo de que los cambios interfieran unos con otros (por ejemplo, pruebas A/B simultáneas). Para mitigar estos riesgos, podemos utilizar salas de chat para anunciar los cambios y encontrar proactivamente los conflictos que puedan existir.

4.4 Revisión y Coordinación

- En el caso de organizaciones más complejas y con arquitecturas más acopladas es posible que tengamos que programar deliberadamente nuestros cambios, y hacer que representantes de los equipos se reúnan, no para autorizar cambios, sino para programar y secuenciar sus cambios para minimizar los accidentes
- Sin embargo, en algunas áreas, como los cambios en la infraestructura global (por ejemplo, los cambios en los nodos la red, cambios en los conmutadores de la red central) siempre tendrán un mayor riesgo asociado. Estos cambios siempre requerirán contramedidas técnicas, como la redundancia la conmutación por error, la realización de pruebas exhaustivas y (en el mejor de los casos) la simulación

4.4 Revisión y Coordinación

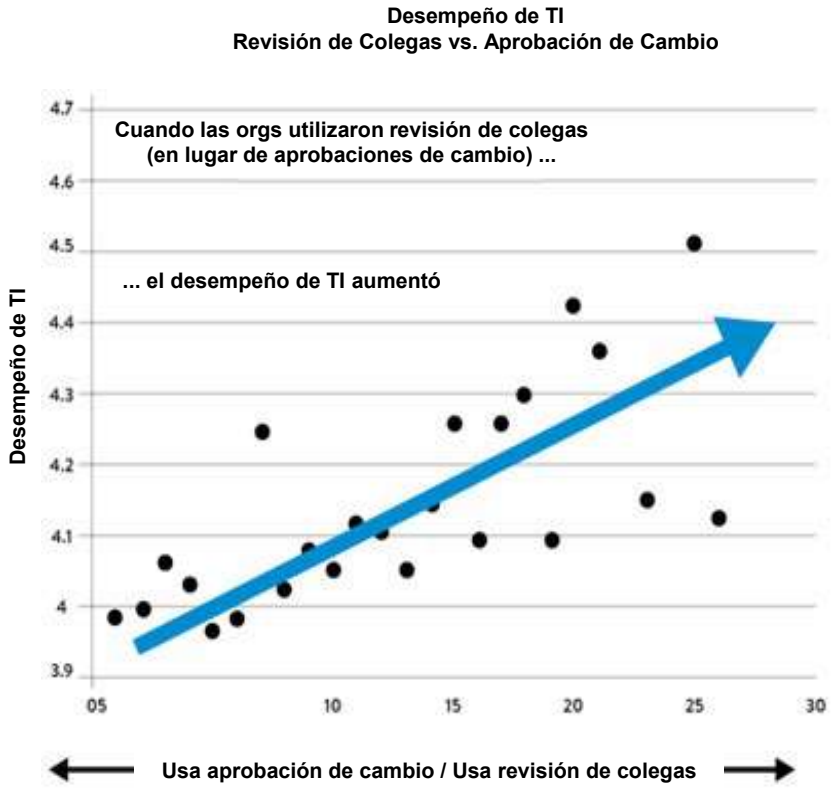
Revisión de Pares del Código

En lugar de exigir la aprobación de un órgano externo antes del despliegue, podemos exigir que los ingenieros hagan revisiones de sus cambios.

El objetivo:

- Encontrar errores, con los ingenieros cercanos al trabajo, examinando nuestros cambios
- Mejorar la calidad de nuestros cambios
- Crear los beneficios de la capacitación cruzada, el aprendizaje entre pares y la mejora de las habilidades

Pero para cambios en la base de datos o componentes esenciales para los negocios con baja cobertura de prueba automatizada, podemos exigir una revisión adicional de un especialista en el tema (por ejemplo, ingeniero de seguridad de la información, ingeniero de base de datos) o varias revisiones (por ejemplo, "+2" en lugar de sólo "+1").



Fuente: DevOps Handbook

4.4 Revisión y Coordinación

El principio de tamaños pequeños de lotes también se aplica a revisiones de código.

- Cuanto mayor sea el tamaño de la alteración que necesita ser revisada, más tiempo lleva a entender y mayor es la carga sobre el ingeniero revisor

Existe una relación no lineal entre el tamaño del cambio y el riesgo potencial de integrar ese cambio - cuando usted pasa de un cambio de código de diez líneas a un código de cien líneas, el riesgo de algo ir mal es más de diez veces mayor, y así sucesivamente.

- La capacidad de criticar significativamente los cambios de código disminuye a medida que el tamaño del cambio aumenta. "Pida a un programador para revisar diez líneas de código, él encontrará diez ediciones. Pídale a él para hacer quinientas palabras, y él dirá que parece bueno".

4.4 Revisión y Coordinación

Directrices generales para las revisiones de código:

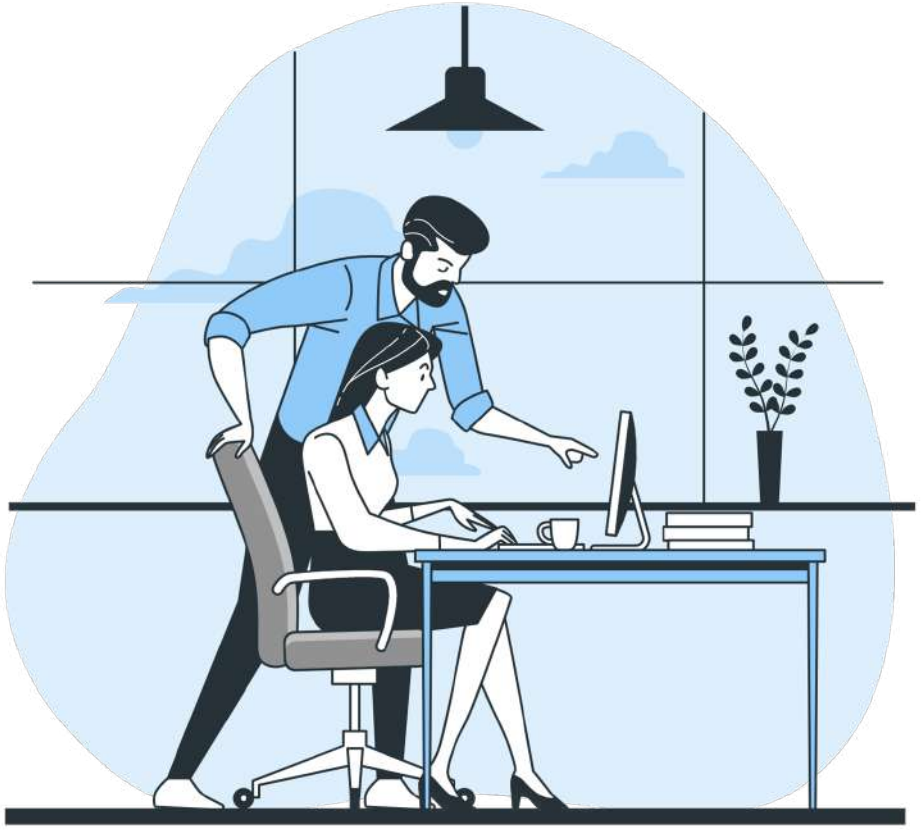
- Todos deben tener a alguien que revise sus cambios antes de integrarlos a la rama principal
- Todos deben supervisar el flujo de commits de sus compañeros de equipo para que se puedan identificar y revisar los posibles conflictos
- Definir qué cambios se consideran de alto riesgo y pueden requerir la revisión de un experto (por ejemplo, cambios en la base de datos, módulos sensibles a la seguridad, como la autenticación, etc.)
- Si alguien envía un cambio que es demasiado grande para entender con facilidad, es decir, no se puede entender su impacto después de leerlo un par de veces, o hay que pedirle al remitente que lo aclare, este cambio debería dividirse en varios cambios más pequeños que puedan entenderse de un vistazo

4.4 Revisión y Coordinación

Programación del Código en Pares

Las revisiones de código vienen en varias formas:

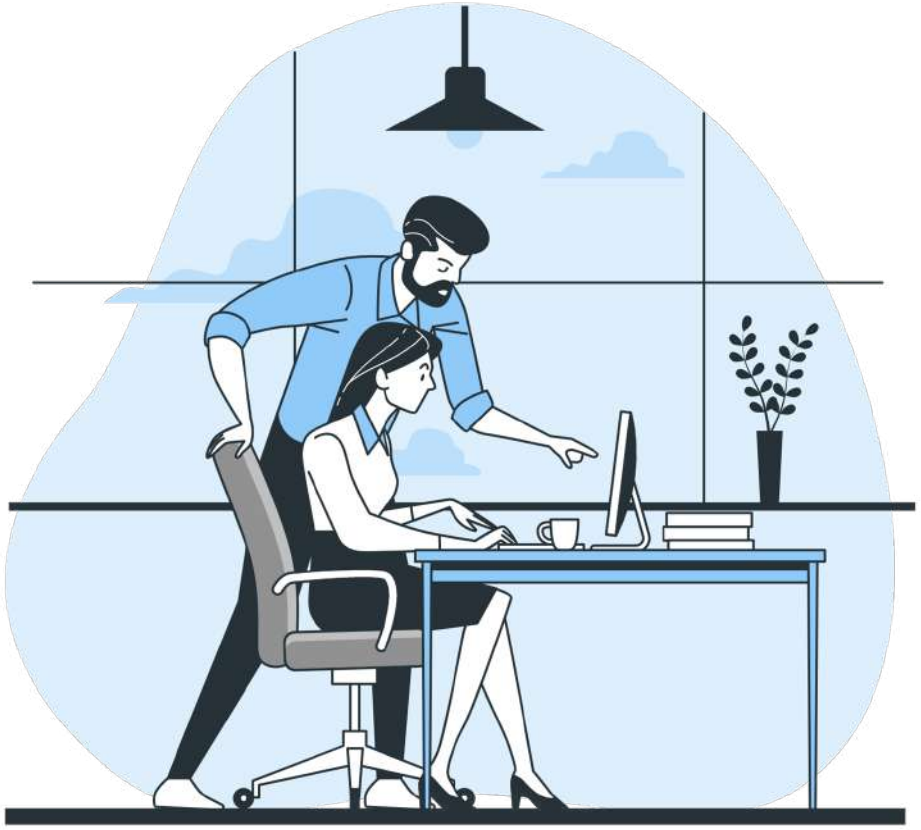
- Programación en pares
- Revisión sobre los hombros
- Envío de correo electrónico
- Revisión de código asistida por herramientas



4.4 Revisión y Coordinación

Resumen

Crear las condiciones que permitan a los ejecutores del cambio apropiarse plenamente de la calidad de sus cambios es una parte esencial de la cultura generativa de alta confianza que estamos tratando de construir. Además, estas condiciones nos permiten crear un sistema de trabajo cada vez más seguro, en el que todos nos ayudamos mutuamente a alcanzar nuestros objetivos, traspasando los límites necesarios para conseguirlo.



Discusión en Grupos – Code Reviews

Parte 5: Aprendizaje y Experimentación Continua

5.0 Las Prácticas Técnicas del Aprendizaje

¿Por qué la Tercera Forma?

Revisaremos las prácticas que crean oportunidades de aprendizaje, de la manera más rápida, frecuente, barata y pronta posible. Esto incluye la creación de aprendizajes a partir de accidentes y fracasos, que son inevitables en sistemas complejos, así cómo prácticas para que estemos constantemente experimentando, aprendiendo, y creando sistemas más seguros. El resultado es una mayor resiliencia y un conocimiento colectivo cada vez mayor.

Los rituales que aumentan la seguridad, la mejora continua y el aprendizaje son:

- Establecer una cultura justa para hacer posible la seguridad
- Inyectar fallos de producción para crear resiliencia
- Convertir los descubrimientos locales en mejoras globales
- Reservar tiempo para crear mejoras organizativas y aprendizaje

También crearemos mecanismos para que cualquier nuevo aprendizaje generado localmente pueda utilizarse rápidamente en toda la organización de manera global. De este modo, también creamos una cultura de trabajo más segura y resistente, de la que las personas están encantadas de formar parte y que les ayuda a alcanzar su máximo potencial.

Cultura del Aprendizaje

Para aumentar la seguridad en sistemas complejos, las organizaciones deben:

- Ser mejores en autodiagnóstico y autodesarrollo
- Necesitan tener capacidad para detectar problemas, resolverlos y multiplicar los efectos, poniendo a disposición las soluciones en toda la organización

Esto crea un sistema dinámico de aprendizaje que nos permite entender nuestros errores y traducir ese entendimiento en acciones que evitan que esos errores se repitan en el futuro.

La historia también muestra cómo las organizaciones que aprenden piensan en fracasos, accidentes y errores, como una oportunidad para aprender y no algo para ser castigado.

¿Cómo crear un sistema de aprendizaje y cómo establecer una cultura justa, además de cómo ensayar rutinariamente y deliberadamente, creando fallas para acelerar el aprendizaje?

Cultura del Aprendizaje

Uno de los prerrequisitos para una cultura de aprendizaje es que, cuando ocurren accidentes (que sin duda ocurrirán), la respuesta a estos accidentes es vista como "justa".

- Cuando se consideran injustas:
  - ✓ Puede impedir investigaciones de seguridad
  - ✓ Promoviendo el miedo
  - ✓ Tornando las organizaciones más burocráticas que más cuidadosas
  - ✓ Cultivando secreto profesional, evasión y autoprotección
- Cuando los ingenieros cometen errores y se sienten seguros al dar detalles sobre ellos:
  - ✓ Están dispuestos a ser responsabilizados
  - ✓ También están entusiasmados en ayudar al resto de la empresa a evitar el mismo error en el futuro

Esto es lo que crea el aprendizaje organizacional.

Cultura del Aprendizaje: Modelos de Tipología Organizacional

Burocrática

Patológica

Nosotros no cometemos errores y no toleramos a la gente que los comete

Creemos una nueva regla

Generativa

Entendamos por qué ocurrió el error

El modelo de tipología organizacional de Westrum: cómo procesan la información las organizaciones (Fuente: Ron Westrum, “A tipology of organisational culture”), BMJ Quality & Safety 13, n.º 2 (2004), doi:10.1136/qshc.2003.009522).

Cultura del Aprendizaje: Modelos de Tipología Organizacional

Patológica	Burocrática	Generativa
La información es **ocultada**	La información podría ser **ignorada**	La información es activamente **buscada**
Los mensajeros son **culpados**	Los mensajeros son **tolerados**	Los mensajeros son **entrenados**
Responsabilidades **eludidas**	**Limitación** de responsabilidades	Responsabilidades son **compartidas**
Se **desaconseja** las conciliaciones entre grupos	Se **tolera** las conciliaciones entre grupos	Se **incentiva** las conciliaciones entre grupos
El fracaso es **ocultado o** genera **chivos expiatorios**	El fracaso conduce a la **justicia** hacia las personas	El fracaso genera **preguntas (causas raíces)**
Las nuevas ideas son **suprimidas**	Las nuevas ideas **crean problemas**	Las nuevas ideas son **bienvenidas**
El modelo de tipología organizacional de Westrum: cómo procesan la información las organizaciones (Fuente: Ron Westrum, "A tipology of organisational culture"), BMJ Quality & Safety 13, n.º 2 (2004), doi:10.1136/qshc.2003.009522).		

Cultura del Aprendizaje: Modelos de Tipología Organizacional

La cultura tiene que ver con rituales diarios que influyen en cómo funciona un equipo

En función de estos aspectos, aquí hay algunas prácticas que puedes implementar para mejorar tu cultura:

- **Incentiva la Colaboración entre miembros y grupos**
- **Capacita y premia a los mensajeros**
- **Fomenta los Riesgos y Responsabilidades compartidas**
- **Busca conectar los sistemas aislados en la organización**
- **Permite que los errores generen preguntas**
- **Apoya la experimentación y el aprendizaje estos llevan a la innovación**

Permitir e Inyectar el Aprendizaje en el Día a Día

¿Qué prácticas crean oportunidades de aprendizaje con rapidez, frecuencia y bajo costo?

- Establecer una cultura de aprendizaje justa
- Programar reuniones post-mortem sin culpa después de los accidentes
- Publicar nuestras autopsias lo más ampliamente posible
- Disminuir las tolerancias de los incidentes para encontrar señales de fallo cada vez más débiles
- Redefinir el fracaso y fomentar la asunción de riesgos calculados
- Inyectar fallos de producción para permitir la resiliencia y el aprendizaje
- Instituir días de juego para ensayar fallos

Los resultados incluyen mayor resiliencia y un conocimiento colectivo cada vez mayor.

Acciones para aumentar la seguridad, la mejora continua y el aprendizaje:

- *Establecer una cultura justa para hacer posible la seguridad*
- *Inyectar fallas de producción para crear resiliencia*
- *Convertir descubrimientos locales en mejoras globales*
- *Reservar tiempo para crear mejoras en la organización y el aprendizaje*

Cultura del Aprendizaje: Modelos de Tipología Organizacional

Programación de post mortem libres de culpa tras incidentes

Post-mortem sin culpa, "los errores de una forma que enfoca los aspectos situacionales del mecanismo de una falla y el proceso de toma de decisión de los individuos cercanos a las fallas".

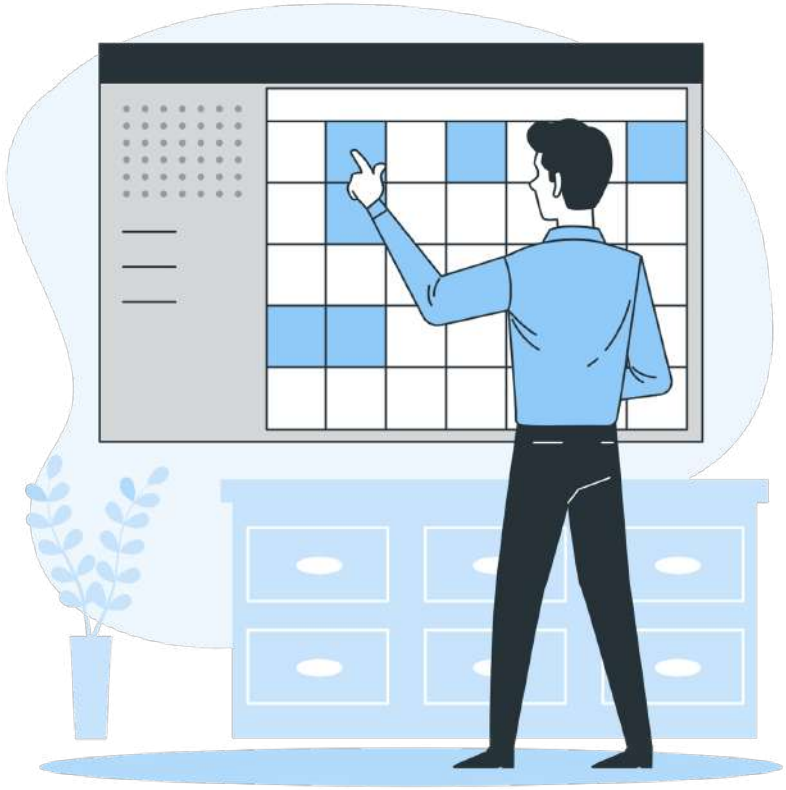
Se programa el post-mortem lo más rápido posible después de la ocurrencia del accidente y antes de que los recuerdos y los eslabones entre la causa y el efecto desaparezcan o las circunstancias cambien.

Los objetivos de una revisión post mortem son muy simples:

- *Identificar lo que ha hecho bien, para que pueda experimentarlo de nuevo*
- *Observar lo que debería haber sido hecho de manera diferente, para que usted pueda refinar*
- *Observar lo que ha hecho mal y sugerir enfoques alternativos*

Agenda

- 5.1 Inyectar fallos de producción para crear resiliencia
- 5.2 Convertir los descubrimientos locales en mejoras globales
- 5.3 Reservar tiempo para crear mejoras organizativas y aprendizaje



5.1 Crear Resiliencia

El Ejército Simio

Netflix tuvo varias discusiones sobre la ingeniería de sus sistemas para manejar automáticamente las fallas. Estas discusiones evolucionaron hacia un servicio llamado Chaos Monkey.

- El Mono del Caos (Chaos Monkey)
- El Gorila del Caos (Chaos Gorilla)
- Caos Kong (Chaos Kong)

https://netflix.github.io/chaosmonkey/

5.1 Crear Resiliencia

El ejército Simio ahora incluye:

- Mono de Latencia: (Latency Monkey)
- Mono de Conformidad(Conformity Monkey)
- Mono Doctor : (Doctor Monkey)
- Mono Conserje: (Janitor Monkey)
- Mono de Seguridad: (Security Monkey)

Fuente: https://blog.aspiresys.pl/technology/chaos-monkey-how-netflix-deals-with-resilience/

5.1 Crear Resiliencia

Programación de post mortem libres de culpa tras incidentes

En la incontestable reunión post-mortem, haremos lo siguiente:

- Construir una línea del tiempo
- Estimular para mejorar la seguridad
- Estimular y fomentar a las personas que cometen errores a ser especialistas
- Aceptar que siempre hay un espacio discrecional donde los seres humanos pueden decidir actuar o no, y que el juicio de esas decisiones está en retrospectiva
- Proponer contramedidas para evitar que un accidente similar ocurra en el futuro

En la reunión, debemos reservar tiempo suficiente para reflexionar y decidir qué contramedidas serán implementadas.

Una vez que las contramedidas hayan sido identificadas, deben ser priorizadas y dadas a un propietario y debe haber cronograma para la implementación.

Esto demuestra que valoramos la mejora de nuestro trabajo diario más que el propio trabajo diario.

5.1 Crear Resiliencia

¿Quién debe estar presente en la reunión?:

- Las personas implicadas en las decisiones que pueden haber contribuido al problema
- Las personas que identificaron el problema
- Las personas que respondieron al problema
- Las personas que diagnosticaron el problema
- Las personas afectadas por el problema
- Y cualquier otra persona que esté interesada en participar en la reunión

5.1 Crear Resiliencia

Introducir fallas en producción para aumentar la resiliencia y el aprendizaje

La resiliencia requiere que primero definamos nuestros modos de fallo y, a continuación, que realicemos pruebas para garantizar que estos modos de fallo funcionen según lo proyectado.

Una manera de hacer esto es inyectando fallas en nuestro entorno de producción y ensayando fallas a gran escala, por lo que estamos seguros de que podemos recuperarnos de accidentes cuando se producen, de preferencia sin afectar a nuestros clientes.

https://netflix.github.io/chaosmonkey/

5.1 Crear Resiliencia

Etapas Base	1. Agile	2. Version Control	3. Continuous Integration CI	4. Continuous Testing	5. Continuous Delivery CD
Etapas Intermedias	6. Continuous Deployment CI/CD	7. Infrastructure as Code	8. Configuration Management	9. Monitoring and Logging	10. Dashboard
Etapas Avanzadas	11. Automated Recovery	12. Automated Environment De-Provisioning	13. Testing in Production	14. User Telemetry	15. Fault Injection

5.1 Crear Resiliencia

Instituir el Día del Juego (Game Day)

El concepto de Game Days viene de la disciplina de ingeniería de resiliencia.

Debemos garantizar que los servicios continúen operando cuando ocurren fallas, potencialmente en todo nuestro sistema, idealmente sin crisis o incluso intervención manual.

"Un servicio no es realmente probado hasta que se rompe en producción".

"Un ejercicio diseñado para aumentar la resistencia a través de la inyección de fallas a gran escala en sistemas críticos".

5.1 Crear Resiliencia

El objetivo del Game Day es ayudar a los equipos a simular y ensayar accidentes para que puedan practicar.

- Programamos un evento catastrófico, para suceder en algún momento en el futuro
- Damos a los equipos tiempo para preparar, eliminar todos los puntos únicos de fallo y crear los procedimientos de monitoreo necesarios, procedimientos de conmutación por error, etc

Nuestro equipo del Día del Juego define y ejecuta ejercicios, como:

- Conducir failovers de base de datos (es decir, asegurarse de que la base de datos secundaria funciona)
- Desactivar una conexión de red importante para exponer problemas en procesos definidos

Cualquier problema o dificultad encontrada es identificado, corregido y probado de nuevo.

5.1 Crear Resiliencia

Al ejecutar Game Day, progresivamente:

- Creamos un servicio más resiliente
- Un mayor grado de garantía de que podemos reanudar las operaciones cuando ocurriesen eventos inoportunos
- Crear más aprendizajes y una organización más resiliente

Al crear fallos en una situación controlada, podemos practicar y crear los manuales que necesitamos.

"Siempre que usted planea proyectar un sistema a escala, lo mejor que puede esperar es construir una plataforma de software confiable sobre componentes que no sean de confianza. Esto le pone en un entorno donde fallas complejas son inevitables e imprevisibles".

5.2 Descubrimientos Locales, Mejoras Globales

¿Cuáles mecanismos posibilitan que nuevos aprendizajes y mejoras descubiertas localmente sean capturados y compartidos globalmente en toda la organización, multiplicando el efecto del conocimiento y de la mejora globales?

- Utilice salas de chat y bots de chat para automatizar y capturar el conocimiento de la organización
- Automatizar los procesos estandarizados en el software para su reutilización
- Crear un único repositorio de código fuente compartido para toda la organización
- Difundir el conocimiento mediante el uso de pruebas automatizadas como documentación y comunidades de práctica
- Diseñar para las operaciones mediante requisitos no funcionales codificados
- Incorporar al desarrollo historias de usuario de operaciones reutilizables
- Garantizar que las opciones tecnológicas ayuden a alcanzar los objetivos de la organización

Elevamos el estado de la práctica de toda la organización para que todos los que trabajan se beneficien de la experiencia acumulativa de la organización.

5.2 Descubrimientos Locales, Mejoras Globales

Diseñar operaciones a través de requisitos no funcionales codificados

La implementación de estos requisitos no funcionales permitirá que nuestros servicios sean fáciles de desplegar y continuar ejecutándose en la producción.

Y que podamos detectar y corregir problemas rápidamente y garantizar que se degrada normalmente cuando los componentes fallan.

Ejemplos de requisitos no funcionales incluyen garantizar que tenemos:

- *Telemetría de producción*
- *La capacidad de acompañar con precisión las dependencias*
- *Servicios que son resilientes*
- *Compatibilidad con versiones anteriores y futuras*
- *La capacidad de archivar datos para administrar el tamaño del conjunto de datos de producción*
- *La capacidad de entender fácilmente los mensajes de log entre los servicios*
- *La capacidad de rastrear solicitudes de usuarios a través de varios servicios*
- *Configuración simple y centralizada de Tiempo de Entrega*

5.2 Descubrimientos Locales, Mejoras Globales

Construir historias de Usuario de Operaciones reutilizables en el Desarrollo

Cuando hay un trabajo de Operaciones que no puede ser totalmente automatizado o transformado en autoservicio, nuestro objetivo es hacer que este trabajo recurrente sea lo más repetitivo y determinista posible.

- Hacemos esto estandarizando el trabajo necesario, automatizando lo máximo posible y documentando nuestro trabajo, para que podamos capacitar mejor a los equipos de producto a planificar y proporcionar recursos para esa actividad
- En lugar de crear manualmente los servidores y, a continuación, ponerlos en producción de acuerdo con las listas de comprobación manual, debemos automatizar el máximo posible este trabajo

Debemos definir colectivamente las transferencias de la forma más clara posible para reducir los tiempos de espera y los errores.

5.2 Descubrimientos Locales, Mejoras Globales

Idealmente, para todos nuestros trabajos recurrentes de Ops, sabremos lo siguiente:

- Qué trabajo es necesario, quién es necesario para ejecutarlo, cuáles son las etapas para concluirlo y así sucesivamente
- Por ejemplo, "Sabemos que un lanzamiento de alta disponibilidad lleva catorce etapas, exigiendo trabajo de cuatro equipos diferentes y, en las últimas cinco veces en que hicimos eso, tomó un promedio de tres días".

"Historias de usuario del Ops" bien definidas que representan actividades de trabajo que pueden ser reutilizadas en todos nuestros proyectos (por ejemplo, despliegue, capacidad, seguridad, etc.).

Exponemos el trabajo repetitivo de las operaciones de TI de una manera que aparece junto al trabajo de desarrollo, permitiendo una mejor planificación y resultados más repetitivos.

5.2 Descubrimientos Locales, Mejoras Globales

Repositorio único con código compartido

Un repositorio de código fuente compartido en toda la empresa.

Al actualizar cualquier elemento en el repositorio de código fuente (por ejemplo, una biblioteca compartida):

- Propagación rápida y automática a todos los demás servicios que utilizan esta biblioteca y se integra a través del pipeline de despliegues de cada equipo

El repositorio de código fuente compartido es uno de los mecanismos más poderosos utilizados para integrar los descubrimientos locales en toda la organización.

5.2 Descubrimientos Locales, Mejoras Globales

En el repositorio de código fuente compartido también se colocan otros artefactos que codifican el conocimiento y el aprendizaje, incluyendo:

- Estándares de configuración para nuestras bibliotecas, infraestructura y entornos (recetas de chef, manifiestos de títeres, etc.)
- Herramientas de despliegue
- Prueba de estándares y herramientas, incluso la seguridad
- Herramientas de pipeline de despliegue
- Herramientas de monitoreo y análisis
- Tutoriales y estándares

Ventajas:

- *Permite a los usuarios acceder a todo el código actualizada, sin necesidad de coordinación*
- *Uno de los mecanismos más poderosos que tenemos para propagar conocimiento*

5.2 Descubrimientos Locales, Mejoras Globales

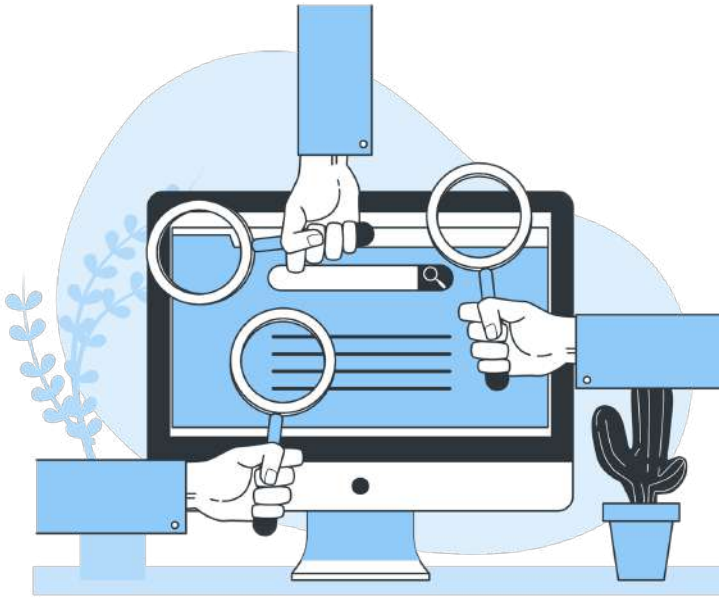
Automatizar procesos estandarizados

En lugar de poner nuestros conocimientos en documentos de Word:

- Transformarlos en un formato ejecutable que los hace más fáciles de reutilizar

Objetivo:

- Permitir que las prácticas sean ampliamente adoptadas



Codificando nuestros procesos manuales en código que es automatizado y ejecutado, permitimos que el proceso sea ampliamente adoptado, proporcionando valor para cualquiera que los utilice.

5.2 Descubrimientos Locales, Mejoras Globales

Divulgue el conocimiento usando pruebas automatizadas

Garantizar que cada una de estas bibliotecas tenga cantidades significativas de pruebas automatizadas incluidas significa que estas bibliotecas se auto documentan y se muestran a otros ingenieros cómo utilizarlas.

Este beneficio será casi automático si tuviésemos prácticas de desarrollo orientado a pruebas, donde las pruebas automatizadas se escriben antes de escribir el código.

Crear grupos de discusión o salas de chat para cada biblioteca o servicio, para que cualquier persona que tenga dudas pueda recibir respuestas de otros usuarios, que generalmente son más rápidos para responder que los desarrolladores.

5.2 Descubrimientos Locales, Mejoras Globales

Las opciones tecnológicas deben cumplir con los objetivos organizacionales

Para maximizar la productividad del desarrollador, el que se utiliza de arquitecturas orientadas a servicios, pequeños equipos de servicio pueden crear y ejecutar sus servicios en cualquier lenguaje o estructura que mejor cumple a sus necesidades específicas.

En algunos casos, eso es lo que mejor nos permite alcanzar nuestros objetivos organizacionales.

El objetivo es identificar las tecnologías que:

- *Impiden o retrasan el flujo de trabajo*
- *Crean desproporcionadamente altos niveles de trabajo no planificado*
- *Crean desproporcionadamente un gran número de solicitudes de soporte*
- *Son más incoherentes con nuestros resultados arquitectónicos deseados (por ejemplo, tasa de transferencia, estabilidad, seguridad, confiabilidad, continuidad de negocios)*

5.3 Permitir e Inyectar el Aprendizaje en el Día a Día

Otras formas de reservar tiempo para el aprendizaje y la mejora de la organización, institucionalizando aún más la práctica de dedicar tiempo a mejorar el trabajo diario, incluyen:

- Implementar Blitz de Mejora
- Institucionalizar los rituales para saldar la deuda técnica
- Permitir que todos enseñen y aprendan
- Comparta sus experiencias en las conferencias de DevOps
- Cree consultorías y entrenadores internos para difundir las prácticas

El resultado de estas mejoras suele ser un nuevo enfoque para resolver un problema, como nuevas disposiciones de los equipos, nuevos medios de transporte material e información, un espacio de trabajo más organizado o un trabajo estandarizado. También pueden dejar una lista de cambios por hacer en el futuro.

5.3 Permitir e Inyectar el Aprendizaje en el Día a Día

Deuda Técnica

Blitz de mejora (o a veces un blitz kaizen), definida como un período de tiempo dedicado y concentrado para tratar de una cuestión específica, muchas veces a lo largo de varios días.

- Blitz: un grupo se reúne para concentrarse intensamente en un proceso con problemas
- El objetivo es mejorar el proceso a través del uso concentrado de personas de fuera del proceso para aconsejar a aquellos normalmente dentro del proceso. Se puede usar para refactorizar código, migrar código obsoleto, identificar y corregir fallos en la funcionalidad completa, o atacar otros elementos puntuales de deuda técnica

5.3 Permitir e Inyectar el Aprendizaje en el Día a Día

Institucionalice rituales para pagar la deuda técnica

- Una de las maneras más fáciles de hacer esto es programar y realizar actividades diarias o semanales o blitzs de mejora continua
- Ningún trabajo de historias está permitido
- Además de los términos orientados al Lean kaizen blitz y blitz de mejora, la técnica de rituales dedicados al trabajo de mejora también ha sido llamada de limpiezas de primavera u otoño y semanas de inversión de cola

5.3 Permitir e Inyectar el Aprendizaje en el Día a Día

Habilite todos a enseñar y aprender

- Una cultura dinámica de aprendizaje crea condiciones para que todos puedan no sólo aprender, sino también enseñar
- Los asuntos son lo que nuestros colaboradores quieren aprender
- Para todos los profesionales de tecnología que aman innovar, aman el cambio, hay un futuro maravilloso y vibrante delante de nosotros

Enseñar:

- *Métodos didácticos tradicionales (por ejemplo, personas que tienen clases, participando en entrenamientos)*
- *Métodos más experimentales o abiertos (por ejemplo, conferencias, talleres, tutoría)*

5.3 Permitir e Inyectar el Aprendizaje en el Día a Día

Comparta sus experiencias en las Conferencias DevOps

- Para ayudar a construir una organización de aprendizaje, debemos alentar a nuestros ingenieros
- DevOpsDays sigue siendo una de las series de conferencias auto organizadas más vibrantes de la actualidad

Muchas prácticas de DevOps fueron compartidas y promulgadas en eventos. Permaneció libre o casi libre, apoyado por una comunidad vibrante de comunidades y proveedores profesionales.

5.3 Permitir e Inyectar el Aprendizaje en el Día a Día

Cree Consultoría interna y Entrenadores para divulgar prácticas

- Crear una organización interna de consultoría y coaching es un método comúnmente usado para diseminar conocimientos en toda la organización. Esto puede venir de muchas maneras diferentes
- Google utiliza varios mecanismos para impulsar la adopción, pero uno de los más famosos fue el **Boletín en los Baños Públicos**

Prueba en el cuarto de aseo: boletín informativo publicado en casi todos los cuartos de aseo de casi todas las oficinas de Google en todo el mundo.

"El objetivo es aumentar el grado de prueba de conocimiento y sofisticación en toda la empresa. Es cuestionable si una publicación sólo on-line ha implicado a personas en el mismo grado".

5.3 Permitir e Inyectar el Aprendizaje en el Día a Día

Cultura de Aprendizaje: Resumen

Se indicó cómo podemos instituir rituales que ayudan a reforzar la cultura de que todos somos aprendices a lo largo de la vida y que valoramos la mejora del trabajo diario en detrimento del propio trabajo diario.

- Lo hacemos reservando tiempo para pagar la deuda técnica
- Creamos foros que permitan a todos aprender y enseñar unos a otros
- Ofrecemos especialistas para ayudar a los equipos internos

Nos ayudamos unos a otros a alcanzar nuestro pleno potencial como seres humanos.

Discusión en Grupos – State of DevOps Report

Parte 6: Integrando Gestión del Cambio, Seguridad de la Información y Cumplimiento

6.0 Integrando DevOps con Otras Áreas

Puntos a cubrir

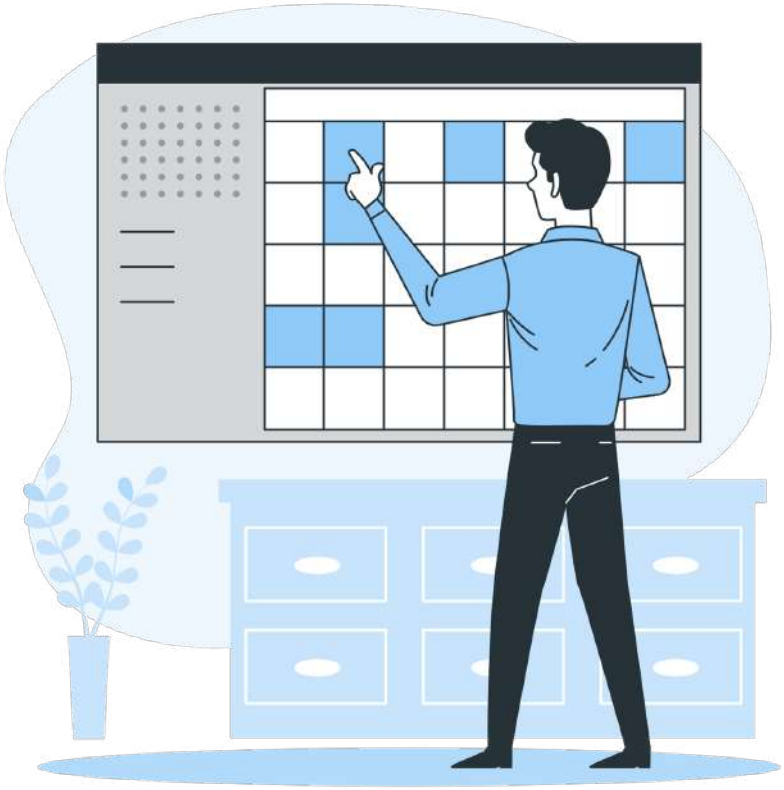
A lo largo de esta sección explicaremos cómo tomar los principios de DevOps y aplicarlos a la seguridad de la información, ayudándonos a alcanzar nuestros objetivos y asegurándonos de que la seguridad forma parte del trabajo de todos, todos los días. Una mejor seguridad garantiza que seamos defendibles y sensatos con nuestros datos, que podamos recuperarnos de los problemas de seguridad antes de que sean catastróficos y, lo más importante, que podamos hacer que la seguridad de nuestros sistemas y datos sea mejor que nunca.

En esta sección explicaremos:

1. Seguridad de la Información somos todos.
2. Integración con Gestión del Cambio, Seguridad y otros requisitos de Conformidad y Cumplimiento

Agenda

- 6.1 Seguridad de la Información somos todos.
- 6.2 Integración con Gestión del Cambio, Seguridad y otros requisitos de Conformidad y Cumplimiento



6.1 Seguridad de la Información Somos Todos

La seguridad de la información como el trabajo de todos, todos los días

Cuando el área de Infosec se organiza como un silo fuera del Desarrollo y de Operaciones, muchos problemas surgen.

Infosec generalmente tiene menor número de colaboradores que las otras áreas, sin automatización e integración de los elementos de seguridad de la información en el trabajo diario de Dev y Ops, Infosec sólo puede realizar la verificación de conformidad, lo que es lo opuesto de la ingeniería de seguridad de la información.

DevOps puede ser una de las mejores maneras de integrar la seguridad de la información al trabajo diario de todos los involucrados en el flujo de valor de tecnología.

6.1 Seguridad de la Información Somos Todos

Prácticas para Seguridad de la Información

- Integrar la seguridad en las demostraciones de iteración del desarrollo
- Integrar la seguridad en el seguimiento de los defectos y en los análisis posteriores
- Integrar los controles de seguridad preventiva en los repositorios de código fuente y servicios compartidos
- Integrar la seguridad en nuestro proceso de despliegue
- Garantizar la seguridad de la aplicación
- Garantizar la seguridad de nuestra cadena de suministro de software
- Garantizar la seguridad del entorno
- Integrar la seguridad de la información en la telemetría de producción
- Crear telemetría de seguridad en nuestras aplicaciones
- Crear telemetría de seguridad en nuestro entorno
- Proteger nuestro pipeline de despliegue

6.1 Seguridad de la Información Somos Todos

Integrar Controles de Seguridad Preventivos

Agregar al repositorio de código fuente compartido:

- **Cualquier mecanismo o herramienta** que nos ayude a garantizar que nuestras aplicaciones y entornos estén seguros
- **Bibliotecas pre-aprobadas por la seguridad** para atender a objetivos específicos de Infosec, como bibliotecas y servicios de autenticación y encriptación

El control de versiones también sirve como un mecanismo de comunicación omnidireccional para mantener a todas las partes conscientes de los cambios que se están haciendo.

6.1 Seguridad de la Información Somos Todos

Integrar Controles de Seguridad Preventivos

El objetivo es proporcionar las bibliotecas o servicios de seguridad que cada aplicación o entorno moderno requiere, como la activación de la autenticación, autorización, administración de contraseñas, cifrado de datos y así sucesivamente.

Además, podemos proporcionar a Dev y Ops configuraciones de seguridad específicas para los componentes que utilizan en sus pilas de aplicaciones, como para el registro, la autenticación y el cifrado.

Podemos incluir elementos como:

- Bibliotecas de código y sus configuraciones recomendadas
- Administración secreta
- Paquetes de SO y compilaciones

6.1 Seguridad de la Información Somos Todos

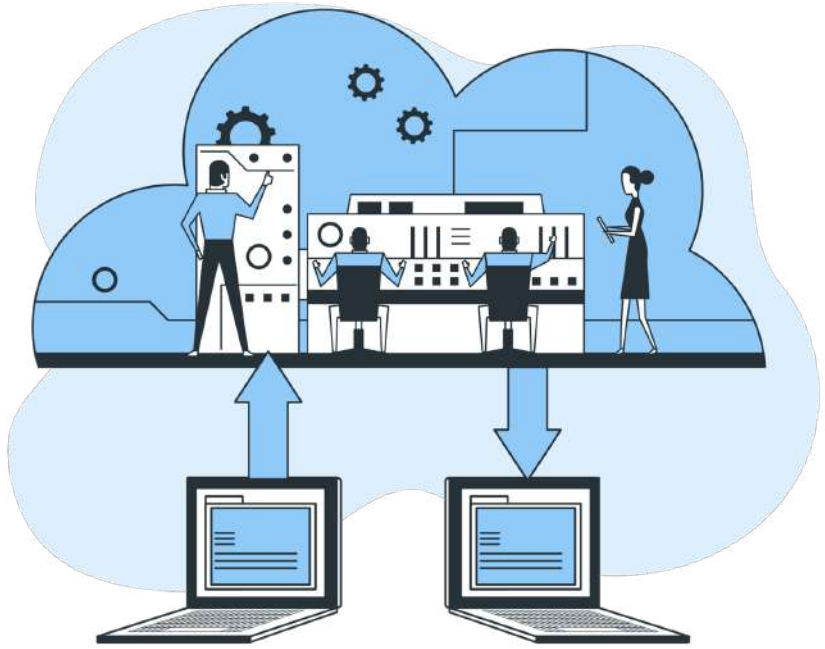
Integrar Seguridad en el pipeline de despliegue

El objetivo:

- Proporcionar a Dev y Ops el feedback rápido sobre su trabajo
- Permitir que detecten y corrijan rápidamente problemas de seguridad como parte de su trabajo diario
- Impulsa el aprendizaje y evita errores futuros

Acciones:

- Automatizar el máximo posible de nuestras pruebas de seguridad
- Pruebas de seguridad automatizadas se ejecutarán en el pipeline de despliegue



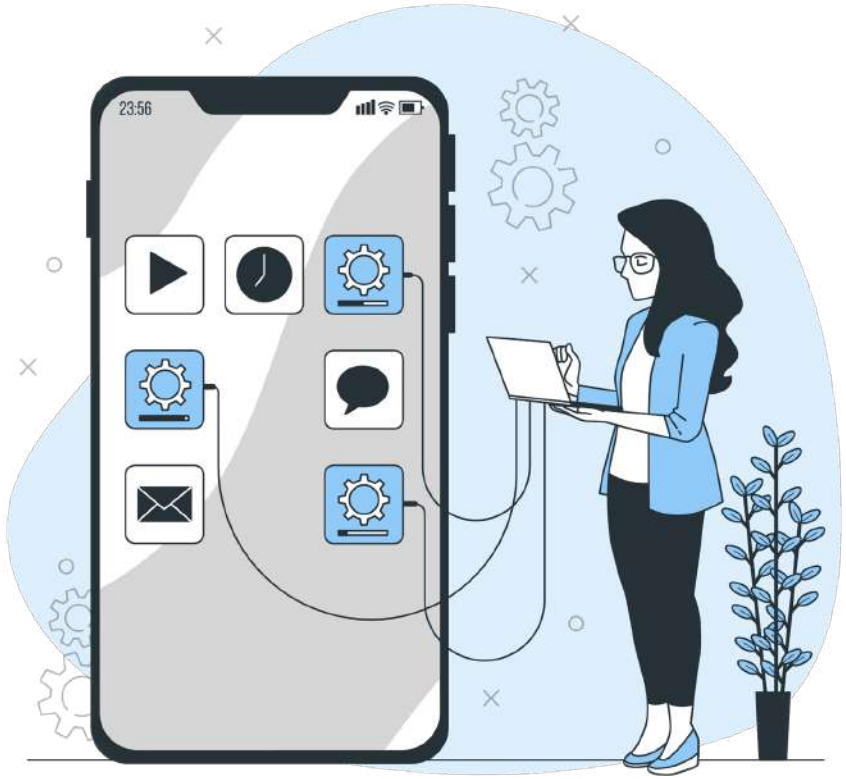
6.1 Seguridad de la Información Somos Todos

Integrar Telemetría en la Seguridad

Para detectar un comportamiento problemático del usuario, que pueda ser un indicador o un facilitador de fraude y acceso no autorizado, debemos crear la telemetría relevante en nuestras aplicaciones.

Los ejemplos pueden incluir:

- Logins de usuario exitosos y fallidos
- La contraseña del usuario se restablece
- La dirección de correo electrónico del usuario se restablece
- Cambios en la tarjeta de crédito del usuario

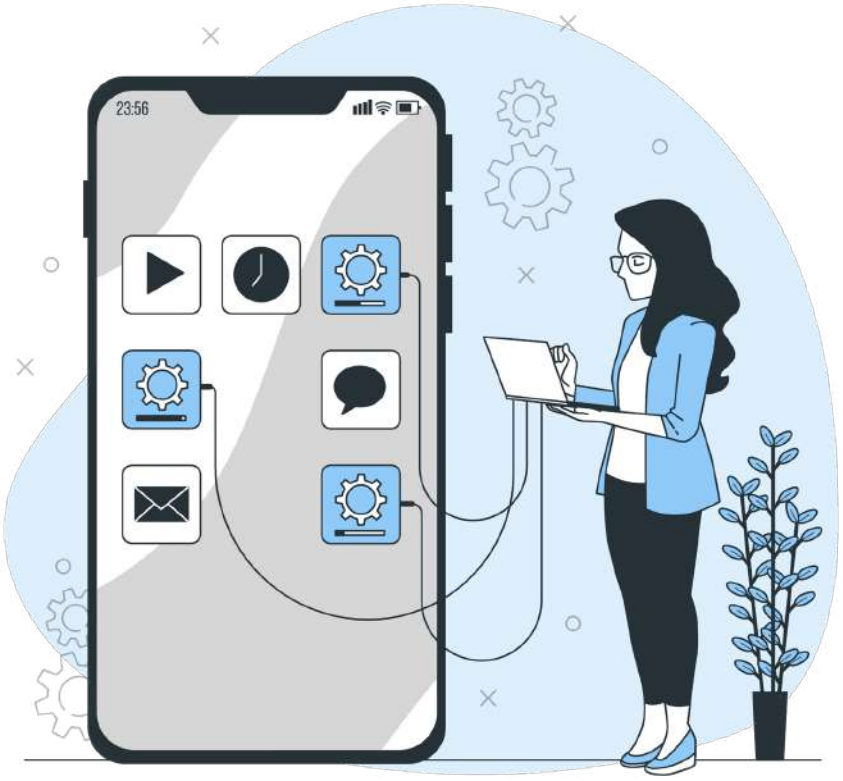


6.1 Seguridad de la Información Somos Todos

Crear telemetría suficiente en nuestros entornos para detectar indicadores precoces de acceso no autorizado, especialmente en los componentes que se están ejecutando en la infraestructura que no controlamos (por ejemplo, entornos de hospedaje en la nube).

Ejemplos de alertas sobre ítems:

- Cambios en el sistema
- Cambios en el grupo de seguridad
- Cambios en la configuración
- Cambios en la infraestructura de la nube
- Intentos de XSS
- Intentos de SQLi
- Errores de servidor Web



6.1 Seguridad de la Información Somos Todos

Proteja el pipeline de despliegue

La infraestructura que soporta nuestros procesos de integración continua y el despliegue continuo también presenta una nueva área de superficie vulnerable a ataques. Por ejemplo, un invasor podría inyectar cambios maliciosos en nuestro repositorio de control de versiones y, por lo tanto, inyectar cambios maliciosos en nuestras aplicaciones y servicios.

6.1 Seguridad de la Información Somos Todos

Para proteger la construcción, integración o el despliegue continuo, las estrategias de mitigación pueden incluir:

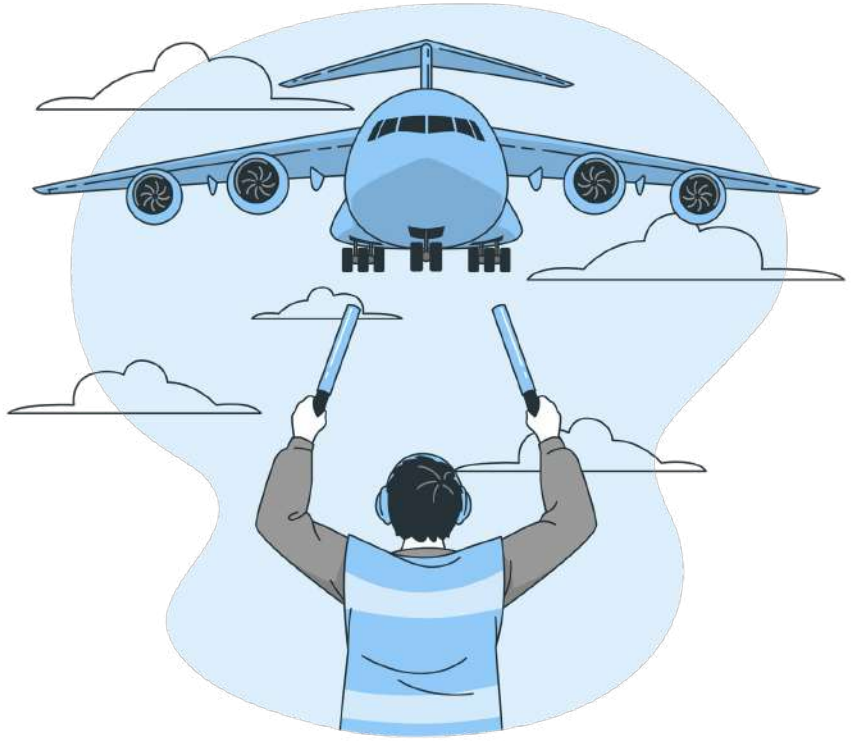
- Proteger servidores de construcción e integración continuos
- Revisar todos los cambios introducidos en el control de versiones
- Detectar cuando el código de prueba contiene llamadas de API sospechosas
- Garantizar que todo proceso de IC se ejecute en su propio contenedor o VM aislada
- Garantizar de que las credenciales de control de versiones utilizadas por el sistema de CI sean de sólo lectura

6.2 Integrando con Otras Áreas

Integración con Gestión del Cambio, Seguridad y otros requisitos de Conformidad y Cumplimiento

En esta sección estudiaremos cómo proteger nuestra tubería de despliegue, así como la forma de lograr los objetivos de seguridad y cumplimiento en nuestros entornos de control, incluyendo la gestión de cambios y la separación de funciones. Algunas buenas prácticas incluyen:

- Integrar la seguridad y el cumplimiento en los procesos de aprobación de cambios
- Recategorizar la mayoría de nuestros cambios de menor riesgo como cambios estándar*
- Definir el tratamiento de los cambios normales*
- Reducir la dependencia en la separación de funciones
- Garantizar la documentación y las pruebas para los auditores y los responsables de cumplimiento



**en base a ITIL*

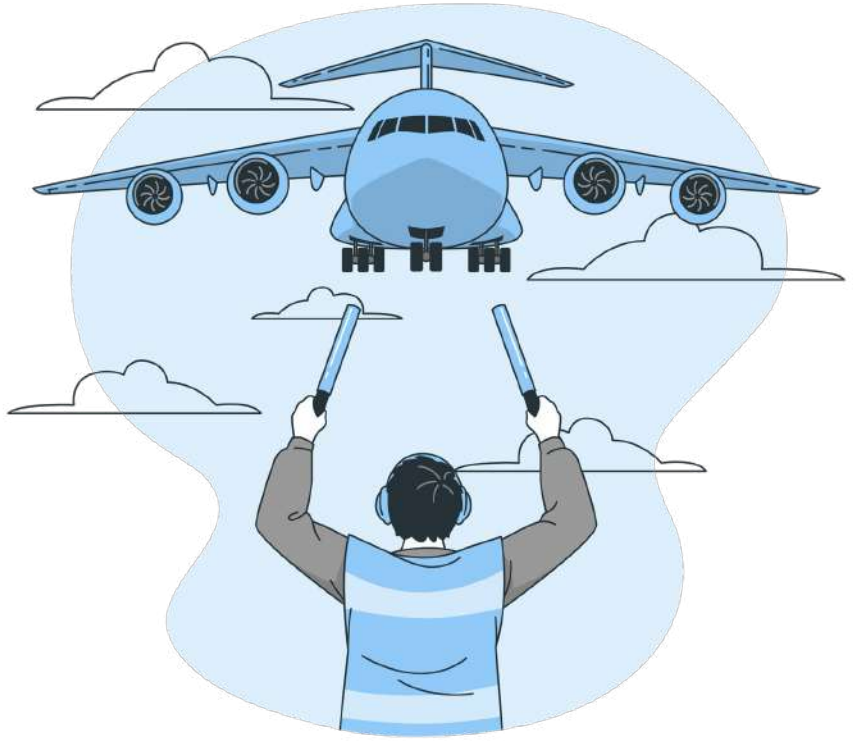
6.2 Integrando con Otras Áreas

Seguridad en el Cambio

Casi toda organización de TI de cualquier tamaño significativo tendrá procesos de gestión de cambios existentes, que son los principales controles para reducir riesgos de operaciones y de seguridad.

El pipeline de despliegues ejecutando implementaciones de bajo riesgo:

- Mayoría de nuestros cambios no necesitará pasar por un proceso manual de aprobación de cambios
- Habremos confiado en controles como pruebas automatizadas y monitoreo de producción proactivo



6.2 Integrando con Otras Áreas

Integrar la seguridad y el cumplimiento en procesos de aprobación de cambios

Los procesos de gestión de cambios buscan implementar controles para reducir los riesgos en las operaciones y de seguridad. Los directores de cumplimiento y de seguridad confían en los procesos de gestión de cambios para cumplir con los requisitos, y suelen exigir pruebas de que todos los cambios han sido debidamente autorizados.

Existen estos tipos de cambios:

- **Cambios estándar:** Son cambios de menor riesgo que siguen un proceso establecido y aprobado, pero que también pueden ser pre aprobados
- **Cambios normales:** Son cambios de mayor riesgo que requieren la revisión o aprobación de la autoridad de cambio acordada
- **Cambios urgentes:** Se trata de cambios de emergencia y, en consecuencia, de alto riesgo potencial, que deben ponerse en producción inmediatamente (por ejemplo, parche de seguridad urgente, restauración del servicio)

6.2 Integrando con Otras Áreas

Nuestro pipeline de despliegue de estar configurado correctamente para que los despliegues sean de bajo riesgo, y la mayoría de nuestros cambios no necesitarán pasar por un proceso manual de aprobación de cambios, porque habremos confiado en controles como las pruebas automatizadas y la supervisión proactiva de la producción.

6.2 Integrando con Otras Áreas

Recategorizar la mayoría de nuestros cambios de menor riesgo como cambios estándar*

Al contar con una cadena de despliegue fiable, ya nos hemos ganado una reputación de despliegue rápido, fiable y sin dramas. En este punto, deberíamos tratar de obtener el acuerdo de Operaciones de que nuestros despliegues son de bajo riesgo y pueden ser clasificados como cambios estándar, pre-aprobados por el CAB.

Lo ideal es que los despliegues sean realizados automáticamente por nuestras herramientas de gestión de la configuración y de canalización del despliegue (por ejemplo, Puppet, Chef, Jenkins) y que podamos vincular automáticamente estos registros de despliegues en PROD a elementos específicos en nuestras herramientas de planificación del trabajo (por ejemplo, JIRA, Rally, LeanKit, ThoughtWorks Mingle), lo que nos permite crear más contexto para nuestros cambios, como la vinculación a los defectos, los incidentes de producción o las historias de usuario.

Al hacer esto, podemos rastrear un despliegue de producción a los cambios en el control de versiones y, a partir de ahí, rastrearlos hasta los tickets de la herramienta de planificación.

6.2 Integrando con Otras Áreas

Qué hacer cuando los cambios son clasificados como cambios normales

Para cambios normales que requerirán la aprobación de al menos un subconjunto de CAB nuestro objetivo es garantizar que podamos desplegar rápidamente, aunque no esté totalmente automatizado.

Por ejemplo, podríamos crear automáticamente un ticket de cambio de ServiceNow con un enlace a la historia de usuario de JIRA, junto con los manifiestos de construcción y los resultados de las pruebas de nuestra y enlaces a los scripts de Puppet/Chef que se ejecutarán. Esto incluye identificar por qué estamos haciendo el cambio (por ejemplo, proporcionando un enlace a las características defectos o incidentes), a quién afecta el cambio y qué se va a cambiar.

Nuestro objetivo es compartir las pruebas y los artefactos que nos dan confianza en que el cambio funcionará en producción tal y como se ha diseñado, para que CAB puede aprobar más rápidamente.

6.2 Integrando con Otras Áreas

Reducir la dependencia de la separación de funciones

A medida que aumenta la complejidad y la frecuencia de los despliegues, la realización de despliegues de producción requiere cada vez más que todos los miembros del flujo de valor vean rápidamente los resultados de sus acciones.

La separación de funciones a menudo puede impedir esto al ralentizar y reducir la información que reciben los ingenieros sobre su trabajo. Esto impide que los ingenieros asuman la calidad de su trabajo y reduce la capacidad de la empresa para crear aprendizaje organizativo.

En consecuencia, siempre que sea posible, debemos evitar utilizar la separación de funciones como control. En su lugar, deberíamos optar por controles como la programación por parejas la inspección continua de las comprobaciones del código y la revisión del mismo.

Estos controles nos pueden dar la seguridad necesaria sobre la calidad de nuestro trabajo.

Siempre que sea posible, debemos evitar el uso de la separación de tareas como control.

6.2 Integrando con Otras Áreas

Garantizar la documentación y las pruebas necesarias para auditores y responsables de cumplimiento

A medida que las organizaciones tecnológicas adoptan cada vez más patrones DevOps, hay más tensión que nunca entre TI, auditoría y los responsables de cumplimiento. Para ayudar a cerrar esa brecha, hace que los equipos trabajen con los auditores en el proceso de diseño del control.

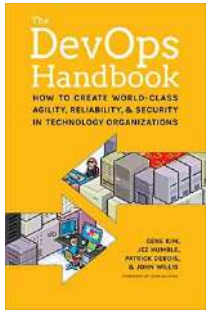
El DevOps Audit Defense Toolkit describe la narración de principio a fin del proceso de cumplimiento y auditoría para una organización ficticia (Parts Unlimited de "The Phoenix Project").
El documento describe cómo podrían diseñarse los controles en una cadena de despliegue para mitigar los riesgos declarados, y proporciona ejemplos de atestados de control y artefactos de control para demostrar la eficacia del control.

Se pretende que sea general para todos los objetivos de control, incluido el apoyo a la presentación de informes financieros precisos, el cumplimiento de la normativa (por ejemplo, SEC SOX-404, HIPAA, FedRAMP, contratos modelo de la UE y la propuesta de normativa Reg-SCI de la SEC), las obligaciones contractuales (por ejemplo, PCI DSS, DOD DISA) y las operaciones eficaces y eficientes.

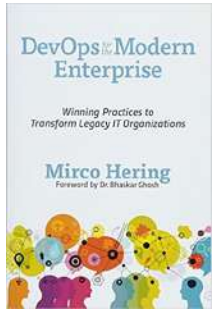
Discusión en Grupos – OWASP

¿PREGUNTAS?
¿DUDAS?
¡CUMPLIDOS!

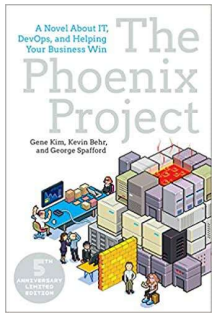
Fuente del Material



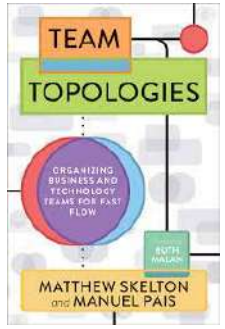
The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations Tapa blanda – Ilustrado, 6 Octubre 2016
de Gene Kim (Author), Patrick Debois (Author), John Willis (Author), Jez Humble (Author), John Allspaw (Foreword)



DevOps For The Modern Enterprise: Winning Practices to Transform Legacy IT Organizations Tapa blanda – 3 Abril 2018
de Mirco Hering (Author), Bhaskar Ghosh (Foreword)



The Phoenix Project (A Novel About IT, DevOps, and Helping Your Business Win) Tapa blanda – Ilustrado, 1 Febrero 2018
de Gene Kim (Author)



Team Topologies: Organizing Business and Technology Teams for Fast Flow Tapa blanda – Ilustrado, 17 Septiembre 2019
de Matthew Skelton (Author), Manuel Pais (Author), Ruth Malan (Foreword)

Fuente del Material

Material de Apoyo

- https://cloud.google.com/devops
- https://web.devopstopologies.com/
- https://itrevolution.com/the-three-ways-principles-underpinning-devops/
- https://techbeacon.com/devops/10-companies-killing-it-devops
- https://techbeacon.com/devops/10-companies-killing-it-devops-2020
- https://github.com/Netflix/SimianArmy
- https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116

Fuente del Material

Ilustraciones:

- Storyset | Customize, animate and download illustration for free
- https://www.pinterest.es/pin/798192733957711009/?d=t&mt=login
- https://www.ithinkupc.com/es/blog/como-conseguir-la-eficiencia-en-un-entorno-bimodal-it
- https://compbcn.es/docker-en-pocas-palabras/
- https://jaxenter.com/devops-shifting-left-172792.html
- https://dev.to/mostlyjason/intro-to-deployment-strategies-blue-green-canary-and-more-3a3
- https://www.boldbi.com/integrations/azure-devops
- https://netflix.github.io/chaosmonkey/
- https://blog.aspiresys.pl/technology/chaos-monkey-how-netflix-deals-with-resilience/

COMPARTE Y VERIFICA TUS LOGROS DE APRENDIZAJE FÁCILMENTE

#DAPC #certiprof

certiprof®

¡Síguenos, ponte en contacto!

www.certiprof.com