Universidad Nacional Pedro Henríquez Ureña
Facultad de Ciencias y Tecnología
Escuela de Informática

Diseño de Guía de Adaptación de Prácticas DevOps para Empresas con Procesos de Ingeniería de Software

Proyecto Final presentado por

Ángel Manuel Adames Montaño

Santo Domingo, D.N.

2022

DEDICATORIAS

A mis padres, Andrés Adames y Rosa Iris Montaño,

Por ser mi mejor ejemplo para seguir, y apoyarme incondicionalmente en todas las etapas de mi vida. Sin ellos no sería quién soy, ni hubiese llegado a donde estoy hoy. Su amor, buenos valores y consejos, son mi mayor tesoro.

A mis hermanas, Rosanna Andreina y Ana Iris,

Por siempre creer en mí y tenerme presente siempre a pesar de la distancia. Me llena de alegría saber que están orgullosas de mí y que puedo servir como su ejemplo a seguir. Sin ustedes muchos de mis logros no tendrían sentido.

A mi mejor amiga, Dahiana Astacio,

Por tu sincero cariño incondicional y la alegría que traes a mi vida. Por estar presente en los momentos donde más lo necesité y ser completamente transparente. Por ti soy mejor persona.

AGRADECIMIENTOS

A Dios,

Por darme la sabiduría y serenidad necesaria para afrontar todos los retos de la vida, por guiarme por el camino correcto y brindarme tu paz en tiempos de debilidad.

A mis padres, Andrés Adames y Rosa Iris Montaño,

Por priorizar mi educación por encima de todas las cosas y enseñarme la importancia de la responsabilidad y de la disciplina.

A mi mejor amiga Dahiana Astacio,

Por ayudarme a mantener mis objetivos de vida claros y motivarme a estar enfocado en lo que debía. Por servir de motivación a terminar mi trayecto académico.

A mis amigos, Bernie Nicasio y Daniel Feliz,

Con ustedes la vida universitaria siempre fue más llevadera. Gracias por llamarme amigo, por aceptarme y respetarme tal cual soy.

A mi mejor amigo, Hermis Botier,

Por ser mi referente y representación más sincera de lo que realmente significa ser un amigo. Por las risas, las largas conversaciones y tu presencia. Me has tratado como un hermano de sangre y así es como te considero.

RESUMEN

En la última década, mucho se ha hablado y expandido sobre lo qué es DevOps y lo que realmente significa. Se han entrevistado y consultado miles de profesionales alrededor de los años tratando de encontrar patrones de éxito comunes que puedan aplicarse o referenciarse de manera genérica y a alto nivel. Por igual, se ha explorado las relaciones existentes entre las operaciones de TI, la ingeniería de software, las prácticas de DevOps, la cultura organizacional, con los objetivos y resultados del negocio.

Con toda esta información, se ha producido una referencia común al éxito de organizaciones que han implementado de manera exitosa las prácticas DevOps y han evidenciado de primera mano las oportunidades que trae sobre la mesa. Sin embargo, aún existe una brecha en cuanto al alcance del movimiento que está pendiente por cubrir. Muchas organizaciones quedan desprotegidas detrás de flujos burocráticos y procesos tradicionales que, si bien han sido funcionales por mucho tiempo, no se adaptan a las necesidades y requerimientos de velocidad y calidad de la industria actual.

Tomando este planteamiento como partida, y apoyándome de la experiencia de primera mano con un rol asociado a estas prácticas, este trabajo de grado pretende incrementar la conciencia sobre DevOps a través de dos grupos esenciales: individuos profesionales en el área de tecnologías de información y organizaciones con procesos de ingeniería de software internos; por lo que una base de conocimiento y una guía de adaptación son los recursos elegidos que podrán promoverse para impulsar el alcance de DevOps actual.

ABSTRACT

In the last decade, much has been said and expanded on what DevOps is and what it really means. Thousands of professionals have been interviewed and consulted over the years trying to find common success patterns that can be applied or referenced generically and at a high level. Likewise, the relationships between IT operations, software engineering, DevOps practices, organizational culture, with business objectives and results have been explored.

With all this information, there has been a common reference to the success of organizations that have successfully implemented DevOps practices and have seen firsthand the opportunities it brings to the table. However, there is still a gap in terms of the scope of the movement that remains to be filled. Many organizations are left unprotected behind bureaucratic flows and traditional processes that, although they have been functional for a long time, do not adapt to the needs and requirements of speed and quality of the current industry.

Taking this approach as a starting point, and relying on first-hand experience with a role associated with these practices, this degree project aims to increase awareness of DevOps through two essential groups: professional individuals in the information technology area and organizations with internal software engineering processes; so a Knowledge Base and Adaptation Guide are the resources of choice that can be leveraged to further the scope of DevOps today.

Contenidos

1. Introducción

Hoy día el desarrollo de software es una parte esencial de las operaciones de negocio de cualquier organización cuyas actividades comerciales se organizan y operan de forma directa, indirecta o parcial, a través de medios digitales. Esto no es una sorpresa, puesto hemos migrado de procesos manuales y sistemas análogos, a flujos totalmente automatizados que han cambiado totalmente nuestro día a día (Smith, 2021).

Aquellas entidades que poseen oferta de servicios o productos no pueden pasar por alto los medios digitales existentes puesto les permiten ampliar su alcance dentro de su propio mercado e industria a usuarios u otras entidades que se manejan exclusivamente a través de estos medios para realizar transacción de información o bienes. Estos medios tecnológicos, tales como *redes sociales*, *páginas y aplicaciones web*; requieren en la mayoría de los casos una gran inversión en términos de tiempo y recursos humanos especializados en la materia, para que su efectividad pueda compensar los resultados comerciales que prometen. Todos estos recursos son sustentados por la ingeniería de software; su importancia e impacto radican en que permite el desarrollo de estos medios tecnológicos reduciendo su complejidad, los costos de construcción, el tiempo de entrega y su general accesibilidad (ExploringBits, 2021).

Para cumplir con estas necesidades, miles de profesionales en tecnologías de información han definido los principios y metodologías necesarias para materializar sistemas funcionales y estables. Sin embargo, los procesos derivados de estos esfuerzos se ven afectados por la costo-efectividad, el error humano, la repetición de tareas monótonas y manuales, y de la débil comunicación entre equipos de diferentes disciplinas.

La actual corriente de empresas que recurren al uso intensivo de software para suplir una necesidad, resolver una problemática, u optimizar procesos existentes; comúnmente llamadas "*Startups*", dejan en clara evidencia que construir software de manera eficaz y oportuna no es una tarea sencilla y no consta con una fórmula mágica para lograr los objetivos que se platean en el modelo de negocios inicial. En la historia del desarrollo de software y su evolución, ha habido significantes propuestas para lograr optimizar estos procesos, sin embargo, en la última década la industria ha sido testigo de una de las que

más éxito ha tenido en corto tiempo: la metodología **ágil (Agile)** y, como un derivado técnico cultural, **DevOps**.

Agile es principalmente una metodología de gestión de proyectos orientada en dos pilares fundamentales: **entrega continua en ciclos cortos, y constante comunicación de cambios**. *Agile* se origina como producto de las deficiencias de la metodología en cascada en la que la gestión de cambios es un proceso complejo de ejecutar correctamente. A su vez, DevOps surge como una respuesta cultural y técnica para mejorar la continuidad de las operaciones inherentes del desarrollo de software enfocándose en la comunicación y entrega de valor de manera práctica y eficiente, apoyándose de la automatización y la reducción de la intervención humana (agilemanifesto.org, n.d.).

Muchos son los beneficios de implementar *Agile* y *DevOps* en los procesos de desarrollo de software de organizaciones de cualquier dimensión, pero se enfrentan al gran reto de la fricción del cambio, de la innovación, y de la adopción de nuevas metodologías que prometen mejoras significativas a cambio del costo operacional de inclusión de nuevas competencias, roles, herramientas, servicios, plataformas y recursos.

Este trabajo de grado intenta atacar esta problemática, llevar a la realidad una guía de adaptación de prácticas *DevOps* a empresas para que su enfoque en la construcción de software sea continuo, moderno y eficiente; así también para diversificar las estrategias que se utilizan para la mejora continua, tanto en aspectos técnicos, comerciales y operacionales.

2. Antecedentes

DevOps ya no es un concepto considerado como "nuevo", puesto desde su concepción original ya han pasado más de 10 años. Sin embargo, nuestra historia empieza en 2008 donde Patrick Debois[1] presentó por primera vez el concepto de *infraestructura ágil* en una conferencia en Toronto llamada **Agile 2008**. En 2009, Patrick organizó la primera conferencia relacionada a DevOps con el nombre de **DevOpsDays**[2], donde el objetivo

[1] Patrick Debois, LinkedIn
[2] devopsdays.org

principal era traer a profesionales de desarrollo de software y operaciones de TI a trabajar en conjunto.

Inicialmente el tema de la conferencia iba a ser *Administración de Sistemas Ágil* ("Agile System Administration" en inglés) pero para Patrick esto era in título muy largo, así que como el enfoque de la conferencia sería orientada a Desarrollo y Operaciones trabajando en conjunto, combinó estas dos palabras obteniendo entonces DevOps. Nunca hubo una gran intención o propósito detrás de dicho nombre, solo se quería un nombre más corto.

De acuerdo con el vídeo *The Short History of DevOps*[3], luego de la primera instancia de *DevOpsDays*, las personas que participaron siguieron conversando sobre este tema en diferentes lugares de la Internet, principalmente en Twitter. Una de las razones del rápido esparcimiento de la palabra DevOps se debió a que en Twitter las discusiones relacionadas a la conferencia fueron etiquetadas con #DevOpsDays, pero debido a las limitantes de caracteres de Twitter por cada tuit en ese entonces, el hashtag evolucionó a ser simplemente #DevOps (Edwards, 2012).

Luego de *DevOpsDays*, se cultivó mucho interés en el tema y tras llevar el mensaje original a varias conferencias más, DevOps ya era algo más que una simple idea y se había convertido en un movimiento emergente. La popularidad de este movimiento no había sido tan rápidamente impulsada de no ser porque en marzo 2011, el analista Cameron Haight de la firma de investigación tecnológica Gartner, predijo que "Para 2015, DevOps evolucionará de una estrategia de nicho empleada por grandes proveedores de nube a una estrategia principal empleada por el 20% de las organizaciones Global 2000" (Haight, 2010). Es difícil afirmar si el análisis de Cameron fue lo que incendió la chispa para que DevOps realmente fuera tan popular, pero sin duda alguna a partir de esta fecha grandes empresas empezaron a interesarse y adoptar DevOps.

DevOps entonces marca desde el 2011 hasta la fecha, una era de transformación y evolución en cuanto a los procesos de ingeniería de software y operaciones tradicionales, trayendo consigo un sin número de beneficios directos e indirectos para todos los niveles de una organización. Es razonable describir DevOps como un viaje, o tal vez una

[3] The Short History of DevOps by Damon Edwards, YouTube

aspiración, en lugar de un destino definido. DevOps busca la mejora continua, mayor rendimiento y eficiencia, despliegues continuos y sobre todo, la entrega continua de valor (Mezak, 2018).

3. Definición del problema

En la actualidad, la mayoría de las empresas cuyas operaciones dependen de la ingeniería de software y no cuentan con los recursos necesarios para adoptar nuevas metodologías que les permitan optimizar la entrega de valor oportuna, permanecen estancadas en términos de **agilidad, innovación** y **adaptación competitiva** cuando se comparan con el mercado en el que se desenvuelven.

DevOps, como un movimiento de transformación cultura y organizacional, promete mejorar considerablemente la comunicación e integración entre equipos, optimizando los procesos existentes y enfocándolos a nuevos paradigmas de operación utilizando los principios de la automatización de procesos, la entrega de valor, la retroalimentación continua, y la acción centrada en objetivos.

Sin embargo, DevOps no es una justa medida, cada organización posee retos y debilidades que son características e inherentes de su industria, cultura y contexto. Para dirigir la adaptación de las prácticas DevOps, se necesita un cambio de paradigma de operación y así mismo un equipo capacitado con la mentalidad y competencias para enfrentar los retos que presentan la resistencia al cambio, la falta de guías, y de expertos en la materia.

Para asistir en la resolución de las problemáticas expuestas, se propone:

- Diseñar una guía de adaptación a alto nivel de prácticas DevOps como soporte evolutivo a empresas e individuos que desconocen las implicaciones técnicas y operativas de dicho movimiento cultural.
- Documentar un currículo de contenido autodidacta e impartido para capacitar profesionales en el área de tecnologías de la información sobre DevOps para que sirvan como expertos en materia de transformación cultural y técnica de las prácticas y principios de DevOps.

4. Justificación

4.1 Originalidad

La cultura y metodología DevOps ha explotado en el mercado internacional en cuanto a posiciones de ingeniería de software se refiere, sin embargo, a pesar de su amplio alcance y reciente corriente de adaptación, la mayoría de proceso de ingenieria en latino américa y el caribe aún no integran estos conocimientos como parte fundamental del ciclo de vida de desarrollo de aplicaciones ni de los procesos organizacionales que dependen de este. Las empresas se quedan estancadas en procesos tradicionales y burocráticos que no se enfocan en las necesidades del equipo y del producto.

Asimismo, los currículos universitarios de educación formal aún no cuentan con materiales, recursos, talleres o asignaturas que incentiven o den a conocer las prácticas que detalla y promueve el movimiento cultural DevOps.

Como contexto técnico, DevOps representa un desafío disciplinario ya que implica dominio de una amplia sombrilla de conocimientos tanto de la parte de desarrollo e ingeniería de software como de operaciones de TI y del negocio. Debido a su complejidad inherente, no existe un plan de carrera formalizado ni sustentando a largo plazo que motive a la masa universitaria y profesional a preferir este rol e impulsar su desarrollo a nivel nacional.

4.2 Profundidad

La guía de adaptación y el contenido educativo propuesto solo plantea una forma óptima de entender todas las implicaciones del proceso de adaptación, siendo una plantilla general que no establezca parámetros específicos para abarcar múltiples necesidades. Los parámetros, métricas o indicadores seleccionados para medir la madurez de adaptación DevOps serán sugeridos de forma tal que pueden ser ajustados a necesidades reales y prácticas de cada organización.

En otras palabras, la profundidad de la solución sugiere flexibilidad en cuanto a su implementación, y solo incita al cambio de ejecución de procesos con soporte de automatizaciones, comunicación continua y enfoque absoluto en la entrega de valor a través de recursos humanos y su rol en el ciclo de desarrollo.

4.3 Impacto

Motivando la masa estudiantil y profesional enfocada en roles tecnológicos a una nueva rama de conocimientos y estrategias modernas basados en metodologías ágiles y comunicación, aumentaría considerablemente la oferta y demanda del mercado local de profesionales especializados en ingeniería DevOps. A su vez, República Dominicana se posicionaría con un líder innovador en la región de Latino América por promover la adaptación de conceptos y metodologías frecuentemente buscadas por la industria y el mercado internacional.

De igual manera, más empresas de todos los niveles e industrias adoptarán nuevos enfoques de trabajo para mejorar el ciclo de vida de entrega de valor de sus objetivos de negocio puesto la transformación empieza de abajo hacia arriba, a diferencia de los enfoques tradicionales que empiezan a definir objetivos muy alejados de la realidad cultural de la organización.

5. Marco teórico

A fines de brindar soluciones con resultados directos e indirectos a las problemáticas ya planteadas, es necesario listar y describir los recursos, procesos, metodologías y tecnologías a utilizar. Debido a que DevOps, a pesar de pretender ser un agente de cambio de cultura y procesos, posee una complejidad técnica asociada que requiere de un nivel técnico intermedio-avanzado para corroborar las justificaciones que se documentan en lo adelante.

5.1 DevOps

DevOps es la combinación de filosofías, prácticas y herramientas culturales que aumenta la capacidad de una organización para entregar aplicaciones y servicios (valor) a alta velocidad (Loukides, 2012). Esto se logra evolucionando y mejorando de manera continua el ciclo de vida de desarrollo de software tradicional y los procesos de gestión de operaciones e infraestructura. La etimología de la palabra DevOps surge de dos componentes principales: Desarrollo de Software "Dev" y Operaciones de TI "Ops".

5.1.1 Ingeniería de Software (Desarrollo de Software)

La Ingeniería de Software es una de las ramas de las ciencias de la computación que estudia la creación de software confiable y de calidad, basándose en métodos y técnicas de ingeniería, y brindando soporte operacional y de mantenimiento.

Dicho de otra forma, se define como un proceso de análisis de los requisitos del usuario y luego el diseño, construcción y prueba de la aplicación de software que satisfará esos requisitos (Martin, 2022).

5.1.2 Operaciones de TI

Las operaciones de TI hacen referencia a los procesos, prácticas e incluso servicios de TI necesarios para satisfacer las necesidades comerciales de los usuarios internos y externos (Raza, 2021). Dicho de otra manera, es también una disciplina que busca integrar eficientemente tres factores: **personas, procesos** y **tecnología**, con el propósito de que estos estén alineados con los objetivos de negocio de la compañía.

Gracias a Operaciones, cada empresa puede aprovechar al máximo sus recursos tecnológicos y enfocar las operaciones de TI en sus necesidades, obteniendo beneficios como el aumento de la productividad, la disminución de costos, el robustecimiento de la seguridad de la información, la optimización de procesos, la identificación de nuevas oportunidades y el mejoramiento de la calidad en la relación con los clientes.

5.2 Metodologías Ágiles (Agile)

Agile es un enfoque iterativo para la gestión de proyectos y el desarrollo de software que ayuda a los equipos a entregar valor a sus clientes de manera continua y en ciclos cortos de tiempo (Altvater, 2017). En lugar de realizar un único lanzamiento de resultados del proyecto, un equipo ágil entrega el trabajo en incrementos pequeños, pero consumibles. Los requisitos, planes y resultados se evalúan continuamente para que los equipos tengan un mecanismo natural para responder al cambio de forma efectiva.

5.2.1 Scrum

Scrum es uno de los marcos de referencia más conocidos de implementación de las metodologías ágiles. En la metodología de Scrum, existen ciclos cortos de entrega

pautados conocidos como Sprint. Un sprint comúnmente puede corresponder desde una semana de trabajo hasta tres semanas, dependiendo de la complejidad y cantidad de requerimientos a los que el equipo se compromete a entregar.

Dentro de Scrum, existen una variedad de procesos pautados con el objetivo general de mantener siempre un canal de comunicación constante para la planificación y ejecución de los cambios. Estos procesos son: **Sprint Planning**, **Daily Scrum**, **Sprint Review**, y finalmente, **Sprint Retrospective**. A estos procesos también se les conoce como eventos de Scrum, o "**Scrum Events**" por su nombre en inglés (ScrumAlliance, s.f.).

5.2.2 Eventos de Scrum

5.2.2.1 Sprint Planning

Es el proceso en el cual el equipo de trabajo estima y prioriza los entregables que se trabajarán como objetivos del siguiente Sprint. Un Sprint no es más que una iteración del ciclo de desarrollo de Scrum delimitada por tiempo: una semana, dos semanas, etc. Como mejor práctica, se recomienda que un Sprint no supere los 21 días luego de iniciado.

5.2.2.2 Refinamiento

El refinamiento (en inglés: "Refinement") es una actividad estrechamente relacionada con la visión del producto, el mercado y el cliente que le gestiona. Por lo tanto, es responsabilidad del propietario del producto; es quién refina el "Product Backlog", utilizando los comentarios de los usuarios y el equipo de desarrollo para ayudar a priorizar y mantener una lista de cosas por hacer apropiadamente documentada.

5.2.2.3 Daily Scrum

Es una reunión diaria idealmente de no más de 15 minutos en la que el equipo sostiene una conversación semi-informal sobre el progreso de los entregables en base a qué se ha hecho, si existe algún bloqueo o si hay cosas que considerar, cambiar o adaptar.

5.2.2.4 Sprint Review

En el evento de Sprint Review el equipo de trabajo se reúne con los dueños del producto para validar lo que se completó durante el Sprint en términos del producto. Se realiza una presentación en la que se muestran los entregables, recibiendo retroalimentación de estos tras validar su funcionamiento y cobertura del alcance esperado. Esta iteración es vital ya que determina en cierta proporción el éxito del Sprint Planning y del trabajo realizado durante ese ciclo.

5.2.2.5 Sprint Retrospective

En el evento de Sprint Retrospective el equipo de trabajo se reúne para validar lo que se completó durante el Sprint en términos de procesos. Se discuten las áreas de mejora para la siguiente iteración, y se toman medidas prácticas para cumplir con dichas mejoras.

5.2.3 Artefactos de Scrum

5.2.3.1 Product Backlog

Es la lista principal de trabajo a realizar. Es una lista dinámica de funciones, requerimientos, mejoras y correcciones que actúa como entrada para el trabajo pendiente del Sprint. En otras palabras, es la lista de "cosas por hacer" del equipo. Es mantenida por el propietario del producto (Product Owner) así que se encarga de revisar, priorizar y actualizar su contenido. A medida que aprendemos más o cambia el mercado, es posible que los requerimientos ya no sean relevantes o que los problemas se resuelvan de otras maneras.

5.2.3.2 Sprint Backlog

Es la lista de elementos, historias de usuarios o correcciones de errores, seleccionados por el equipo de desarrollo para su implementación en el ciclo actual. Antes de cada Sprint, el equipo elige en qué elementos trabajará para el sprint a partir del Product Backlog. Una acumulación de Sprint puede ser flexible y puede evolucionar durante un sprint. Sin embargo, el objetivo fundamental del sprint, lo que el equipo quiere lograr en el sprint actual, no puede verse comprometido.

5.2.3.3 Sprint Goal

También conocido como Incremento. El producto final utilizable de un sprint. Generalmente se muestra el "incremento" durante la demostración de final de sprint. donde el equipo muestra lo que se completó en el sprint. En ágil hay cabida para las variaciones, incluso dentro de los artefactos. Por eso es importante permanecer abierto a la evolución de la forma en que mantiene incluso estos artefactos.

5.2.4 Equipo de Scrum

Scrum se basa encuentra personas, no herramientas. En Scrum, la unidad fundamental de personas, se le llama Equipo Scrum (Scrum Team).

El equipo scrum consta con tres entidades:

- **Scrum Master** (Jefe de Scrum): Es responsable de establecer Scrum como se define en la guía scrum y de la efectividad del Equipo Scrum en su totalidad.
- **Product Owner** (Propietario del Producto): Es responsable de maximizar el valor del producto resultante del trabajo del equipo scrum. La forma en que se hace esto puede variar ampliamente entre organizaciones, equipos Scrum e individuos.
- **Developers** (Desarrolladores): Los desarrolladores son las personas del equipo scrum que se comprometen a crear cualquier aspecto de un Incremento utilizable en cada Sprint.

5.2.5 Kanban

Kanban es otro tipo de implementación de metodologías Ágiles al igual que Scrum. Se caracteriza por ser un método de seguimiento de tareas de forma visual que permite identificar rápidamente cuellos de botella y bloqueos, ya que el flujo de trabajo se representa tal cual en estados y columnas en un tablero y garantiza de forma costo-efectiva el trabajo continuo.

En diferentes escenarios, y bajos condiciones puntuales, es posible ver tanto Kanban como Scrum implementados en un mismo proceso. A este tipo de implementaciones se les suele denominar como "Scrumban", por la combinación de ambos métodos.

5.3 Principios Culturales de DevOps

Los principios de DevOps no son una especificación o estándar, leyes o normativas, más bien, son un marco de referencia. Uno artículo de GitLab menciona que lo importante de DevOps y su metodología son los principios que permiten mejorar las prácticas de desarrollo de software de una organización y destaca entre ellos: automatización del ciclo de vida de desarrollo de software, comunicación y colaboración, mejora continua y minimización de desperdicios y por último el enfoque en las necesidades de los usuarios con lapsos cortos de retroalimentación (GitLab, 2022). A continuación, abundamos un poco más en estos principios de manera resumida.

Otras fuentes afirman que existen 6 principios de DevOps: acción centrada al usuario, crear con un objetivo en mente, responsabilidad de extremo a extremo, equipos autónomos, mejora continua y automatización (DevOps Agile Skills Association (DASA), 2022).

5.3.1 Colaboración

La premisa clave detrás de DevOps es la colaboración entre todos los equipos que forman parte del ciclo de desarrollo, incluyendo, pero no limitándose a desarrollo y operaciones. Esto quiere decir, entre otras cosas, que se trabaja como un solo equipo durante todo el ciclo de vida de desarrollo. Debido a la responsabilidad compartida sobre la entrega de valor al producto, se requiere de un equipo multidisciplinario en donde existan altos niveles de propiedad y frecuente comunicación. La idea principal, en resumen, es colaboración continua desde la idea hasta la entrega.

5.3.2 Automatización

Una práctica esencial de DevOps es automatizar la mayor parte posible del ciclo de vida del desarrollo de software. Esto da al equipo más tiempo en enfocarse en actividades que realmente agregan valor; sobre todo a los desarrolladores, puesto les permite concentrarse en agregar nuevas funcionalidades y corregir bugs.

La automatización es un elemento clave de CI/CD y ayuda a reducir los errores humanos y aumentar la productividad del equipo.

5.3.3 Mejora continua

La mejora continua se estableció como un elemento básico de las prácticas ágiles. Es la práctica de centrarse en la experimentación, minimizar el desperdicio y optimizar la velocidad, el costo y la facilidad de entrega. La mejora continua también está ligada a la entrega continua, lo que permite a los equipos de DevOps impulsar continuamente actualizaciones que mejoran la eficiencia de los sistemas de software.

5.3.4 Acción centrada al cliente

Los equipos de DevOps utilizan cortos ciclos de retroalimentación con clientes y usuarios finales para desarrollar productos y servicios centrados en las necesidades de estos. Las prácticas de DevOps permiten una recopilación y respuesta rápidas a los comentarios de los usuarios mediante el uso de monitoreo en tiempo real y despliegues rápidos y frecuentes. Los equipos obtienen visibilidad inmediata de cómo los usuarios reales interactúan con las aplicaciones y utilizan esa información para trabajar en mejoras adicionales.

5.3.5 Responsabilidad de extremo a extremo

La responsabilidad de extremo a extremo significa que todo el equipo es el dueño del resultado y no hay discriminación ni señalización por culpa. Actuar como equipo y asumir el fracaso y el éxito como una entidad única permite a los colaboradores sentirse parte de una meta mayor que, a su vez, produce altos niveles de propiedad, motivación y resistencia.

No significa que no haya responsabilidades y límites claros, el beneficio real de este principio es garantizar que todo el equipo sea capaz de comprender el impacto de su trabajo en las responsabilidades de los demás; asegurando así que haya una sana rendición de cuentas y transparencia.

5.3.6 Crear con el objetivo en mente

Este principio implica comprender las necesidades de los clientes y crear productos o servicios que resuelvan problemas reales. Los equipos no deben "construir en una burbuja" o crear software basado en suposiciones sobre cómo los consumidores usarán el

software. Más bien, los equipos de DevOps deben tener una comprensión holística del producto, desde la creación hasta la implementación.

5.4 Procesos y Prácticas de DevOps

5.4.1 Integración Continua

La integración continua (Continuous Integration, CI) es la práctica de realizar validaciones frecuentes a los nuevos cambios que se propone agregar a un conjunto de funcionalidades existentes. Los cambios pueden ir desde agregar una nueva funcionalidad nueva, mejorar una existente, corregir una falla identificada previamente o cualquiera otra actividad que implique modificar el código fuente existente de una aplicación o sistema en desarrollo. La integración continua habilita al equipo de desarrolladores, ingenieros de calidad, dueños de producto y gestores de proyecto a tener niveles de confianza más elevados en el proceso de gestión de cambios.

5.4.2 Entrega Continua

Dada la naturaleza de continuidad inherente de los procesos ágiles, la entrega continua (Continuous Delivery, CD) es el conjunto de actividades que se realizan para hacer accesible o visible los cambios agregados en cortos lapsos de tiempo a ambientes controlados de prueba donde las partes interesadas del proceso pueden ver el valor agregado (funcionalidades, correcciones, etc.) de las aplicaciones en desarrollo. La entrega continua también incluye el proceso de llevar estos cambios de ambientes controlados de prueba a ambientes productivos, que es donde los usuarios finales interactúan con el sistema y se genera el real valor organizacional, operativo y de negocio.

5.4.3 Despliegue Continuo

El despliegue continuo lleva la entrega continua a un paso más allá. Con esta práctica, cada cambio que pasa por todas las etapas del ciclo de vida y se entrega al ambiente de producción de manera automática. No hay intervención humana, y solo una prueba fallida evitará que se despliegue un nuevo cambio. Es decir, no existe un proceso manual de aprobación ni nada similar; todo cambio que entra en el ciclo de vida es debido probado y validado y posteriormente lanzado a producción de manera totalmente autónoma.

5.4.4 Pruebas Continuas

Las pruebas continuas son el proceso de ejecutar pruebas automatizadas como parte de la integración continua. Un proceso maduro de pruebas continuas implica de manera implícita que las pruebas también deben de desarrollarse en conjunto con las nuevas funcionalidades garantizando así que en su proceso de integración existen las validaciones apropiadas de lugar. La prioridad de las pruebas automatizadas debe ser siempre enfocada en los flujos críticos para la operación del negocio. La cobertura de las pruebas ayuda a entender qué tan confiables son los nuevos cambios.

Una de las empresas orientadas a proveer este tipo de pruebas lo define como el proceso de ejecutar pruebas automatizadas como parte de la canalización de entrega de software para obtener comentarios sobre los riesgos comerciales asociados con una versión candidata de software lo más rápido posible. Evoluciona y amplía la automatización de pruebas para abordar la mayor complejidad y el ritmo del desarrollo y la entrega de aplicaciones modernas (Tricentis, 2022).

5.4.5 Monitoreo y Alertas

El monitoreo y las alertas son conceptos interrelacionados que juntos forman la base de un sistema de monitoreo. Tienen la capacidad de brindar visibilidad sobre el estado de sus sistemas, ayudar a comprender las tendencias en el uso o el comportamiento y comprender el impacto de los cambios que realiza. Si las métricas se encuentran fuera de los rangos esperados, estos sistemas pueden enviar notificaciones para solicitar a un operador que eche un vistazo, y luego pueden ayudar a mostrar información para ayudar a identificar las posibles causas (DigitalOcean, 2017).

La implementación de monitoreo y alertas como proceso garantiza la observabilidad.

5.4.5.1 Observabilidad

La observabilidad es la capacidad de medir los estados internos de un sistema examinando sus salidas. Un sistema se considera "observable" si el estado actual se puede estimar utilizando solo la información de los resultados, es decir, los datos del sensor. En este contexto, la observabilidad utiliza tres tipos de datos de telemetría (métricas, registros y seguimientos) para proporcionar una visibilidad profunda de los sistemas distribuidos

y permitir que los equipos lleguen a la raíz de una multitud de problemas y mejoren el rendimiento del sistema (Splunk, s.f.).

5.4.6 Seguridad Integrada

La seguridad integrada en DevOps también es también visto mayormente como DevSecOps. Es tomar lo que ya hemos descrito sobre DevOps, pero ahora con un componente adicional, seguridad de la información. DevSecOps significa desarrollo, seguridad y operaciones. Es un enfoque de cultura, automatización y diseño de plataforma que integra la seguridad como una responsabilidad compartida a lo largo de todo el ciclo de vida de TI.

DevSecOps significa pensar en la seguridad de las aplicaciones y la infraestructura desde el principio. También significa automatizar algunas puertas de seguridad para evitar que el flujo de trabajo de DevOps se ralentice. Seleccionar las herramientas adecuadas para integrar continuamente la seguridad, como acordar un entorno de desarrollo integrado (IDE) con características de seguridad, puede ayudar a cumplir estos objetivos.

Sin embargo, la seguridad efectiva de DevOps requiere más que nuevas herramientas: se basa en los cambios culturales de DevOps para integrar el trabajo de los equipos de seguridad más temprano que tarde.

5.5 Pipelines

Los pipelines se pueden definir como un conjunto de pasos que se realizan de forma secuencial o paralela de forma automática tras un evento o suceso particular que sucede en el flujo de desarrollo habitual. Los pipelines permiten abstraer tareas repetitivas, manuales y con tendencias al error humano, de manera tal que el equipo de trabajo se pueda enfocar en la entrega de valor.

5.6 Infraestructura en la nube

5.6.1 La nube

Cuando nos referimos a "la nube" (en inglés, “The Cloud”), hablamos de servidores (computadores) que son accesibles a través de Internet y del software o datos que se alojan en esos servidores, y que son manejados en centros de datos en diferentes partes del

mundo. Si bien el concepto de "la nube" es más frecuente relacionado a servidores gestionados por un proveedor, al cual se le descarga la responsabilidad del provisionamiento y mantenimiento físico de estos centros, es totalmente posible tener una nube propietaria privada.

5.6.1.1 Nube híbrida

El concepto de nube híbrida surge de una infraestructura que posee recursos manejados tanto en la nube como en premisas (dígase, en centro de datos locales o físicos). La nube híbrida es una implementación de la nube muy común para organizaciones con manejo de datos sensitivos de clientes que tiene que regirse por estándares de cumplimiento que le exigen manejar la data de sus clientes directamente y no almacenarla en proveedores externos.

5.6.1.2 Multi-nube

La multinube es un concepto coloquial que se refiere a la implementación de varias nubes. Lo que le diferencia de la nube híbrida es que no necesariamente necesita una nube privada; es posible lograr un esquema multinube al elegir dos proveedores de servicio de nube diferentes, aunque sean públicos.

5.6.2 Servicios en la nube

Los servicios en la nube (en inglés, "Cloud Services") son infraestructuras, plataformas o software alojados por proveedores externos y puestos a disposición de los usuarios a través de Internet. Los servicios en la nube promueven la creación de aplicaciones nativas de la nube. El flujo de datos es mucho más activo, disponible y frecuente debido a que los usuarios pueden conectarse a través de sus clientes de conexión (computadoras, teléfonos inteligentes, etc.) a través de Internet (RedHad, s.f.).

5.6.3 Soluciones como servicio

En el apogeo de la computación y servicios en la nube, y en gran parte gracias a la flexibilidad que permiten, se ha definido una modalidad de negocio e implementación de soluciones que podemos definir a alto nivel como "soluciones como servicio". En las soluciones como servicio existen 4 agrupaciones principales: a) infraestructura como

servicio (IaaS), b) plataforma como servicio (PaaS), c) software como servicio (SaaS) y d) función como servicio (FaaS).

5.6.3.1 Infraestructura como servicio (IaaS)

La infraestructura como servicio (IaaS), también conocida como servicios de infraestructura en la nube, es un tipo de servicio de computación en la nube que ofrece recursos esenciales de cómputo, almacenamiento y redes en demanda, sobre una base de pago por uso. El aspecto más atractivo de IaaS es que permite evitar el costo y la complejidad de comprar y administrar servidores físicos e infraestructura de centros de datos. Cada recurso se ofrece como un componente de servicio separado y solo paga por un recurso en particular durante el tiempo que lo necesite.

5.6.3.2 Plataforma como servicio (PaaS)

La plataforma como servicio (PaaS) es un modelo de computación en la nube en el que un proveedor externo ofrece herramientas de hardware y software a los usuarios a través de Internet. Un proveedor de PaaS aloja el hardware y el software en su propia infraestructura, por lo que el equipo no debe preocuparse de tener que instalar hardware y software internos para desarrollar o ejecutar una nueva aplicación. Al igual que IaaS, PaaS incluye infraestructura (servidores, almacenamiento y redes), pero también middleware, herramientas de desarrollo, servicios de inteligencia comercial (BI), sistemas de administración de bases de datos y más.

5.6.3.3 Software como servicio (SaaS)

El software como servicio (SaaS) es un modelo de distribución de software en el que un proveedor de la nube aloja aplicaciones y las pone a disposición de los usuarios finales a través de Internet. En este modelo, un proveedor de software puede contratar a un proveedor de nube externo para alojar la aplicación. En lugar de instalar y mantener el software, simplemente se accede a él a través de Internet, liberándose de la compleja administración de software y hardware.

5.6.3.4 Función como servicio (FaaS)

FaaS es un tipo de servicio de computación en la nube que permite a los desarrolladores crear, calcular, ejecutar y administrar paquetes de aplicaciones como funciones sin tener que mantener su propia infraestructura.

FaaS es un modelo de ejecución basado en eventos que se ejecuta en contenedores sin estado y esas funciones administran la lógica y el estado del lado del servidor mediante el uso de servicios de un proveedor de FaaS.

Las soluciones FaaS están disponibles en las principales nubes públicas y se pueden aprovisionar en las instalaciones, lo que agrega nuevas capacidades significativas a la TI empresarial para el desarrollo de aplicaciones. Obtenga la guía de estrategia nativa de la nube para prepararse para implementar un enfoque sin servidor con FaaS.

5.7 Proveedores de servicios en la nube

Un proveedor de servicios en la nube (CSP), es una empresa que ofrece soluciones como servicio de computación en la nube: Infraestructura como servicio (IaaS), Software como servicio (SaaS), Plataforma como servicio (PaaS), y/o Función como servicio (FaaS). Los proveedores de servicios en la nube utilizan generalmente sus propios centros de datos para alojar servicios de plataforma e infraestructura basados en la computación en la nube.

Los servicios en la nube generalmente tienen un precio que utiliza varios modelos de suscripción de pago por uso. A los clientes se les cobra solo por los recursos que consumen, como la cantidad de tiempo que se utiliza un servicio o la capacidad de almacenamiento o las máquinas virtuales (VM) utilizadas.

Las plataformas de servicios en la nube más populares son: *Amazon Web Services (AWS)*, *Google Cloud Platform (GCP)* y *Microsoft Azure*.

5.7.1 Amazon Web Services (AWS)

AWS se describe a sí mismo como "Amazon Web Services (AWS) es la plataforma en la nube más completa y ampliamente adoptada del mundo, que ofrece más de 200 servicios completos de centros de datos en todo el mundo. Millones de clientes, incluidas

las empresas emergentes de más rápido crecimiento, las empresas más grandes y las principales agencias gubernamentales, utilizan AWS para reducir costos, volverse más ágiles e innovar más rápido" (AWS, s.f.).

5.7.2 Google Cloud Platform (GCP)

GCP es la oferta de Google para el mercado de computación en la nube como servicio. Muy similar a AWS y Azure, posee una cartera de servicios y recursos que suplen las necesidades de las organizaciones de cualquier tamaño. En su página, se describe de la siguiente manera: "Google Cloud consta de un conjunto de activos físicos, como computadoras y unidades de disco duro, y recursos virtuales, como máquinas virtuales (VM), que se encuentran en los centros de datos de Google en todo el mundo. Cada ubicación del centro de datos está en una región. Las regiones están disponibles en Asia, Australia, Europa, América del Norte y América del Sur. Cada región es una colección de zonas, que están aisladas entre sí dentro de la región" (Google, s.f.).

5.7.3 Microsoft Azure

Azure es la plataforma propietaria de Microsoft para la oferta de soluciones como servicio. Algo que caracteriza a Azure es su amplio enfoque en contextos empresariales y gubernamentales, incluyendo en su oferta una variedad de funcionalidades que soportan las estrategias de licenciamiento y cumplimiento de muchas organizaciones a nivel mundial. De la página oficial: "La plataforma en la nube de Azure consta de más de 200 productos y servicios en la nube diseñados para ayudarlo a dar vida a nuevas soluciones, para resolver los desafíos de hoy y crear el futuro. Cree, ejecute y administre aplicaciones en varias nubes, en las instalaciones y en el perímetro, con las herramientas y los marcos de su elección" (Microsoft Azure, s.f.).

5.8 Tecnologías y recursos

5.8.1 Control de versiones

El control de versiones es la práctica de rastrear y administrar los cambios en el código del software, esto ayuda a los equipos a rastrear cada cambio individual y a tener un historial que se pueda referenciar. El control de versiones es la práctica de rastrear y

administrar los cambios en el código del software, esto ayuda a los equipos a rastrear cada cambio individual y a tener un historial que se pueda referenciar.

En la práctica, el control de versiones se implementa utilizando un sistema de control de versiones (VCS, del inglés "Version Control System").

5.8.1.1 Sistema de control de versiones

Un sistema de control de versiones (VCS) es una utilidad de software que rastrea y administra los cambios en un sistema de archivos. Es lo que permite integrar utilidades de colaboración a la práctica fundamental de control de versiones.

En el ámbito de los archivos de código fuente individuales, un VCS rastreará las adiciones, eliminaciones y modificaciones de las líneas de texto dentro de ese archivo. Uno de los elementos más importantes de un VCS son los repositorios. Los repositorios representan un sistema de archivos manejados por un VCS, usualmente asociados al contexto de una única aplicación o sistema y su código fuente.

En el caso hipotética de que se trabaje, por ejemplo, en dos aplicaciones web. Cada una de estas aplicaciones tendría un repositorio individual donde se administraría su código fuente.

5.8.2 Contenerización

Los contenedores son paquetes livianos del código de la aplicación junto con dependencias, como versiones específicas de lenguajes de programación y librerías necesarias para ejecutar sus servicios de software. Los contenedores son paquetes de software que contienen todos los elementos necesarios para ejecutarse en cualquier entorno. De esta forma, los contenedores virtualizan el sistema operativo y se ejecutan en cualquier lugar, desde un centro de datos privado hasta la nube pública o incluso en la computadora portátil personal de un desarrollador.

El software debe diseñarse y empaquetarse de manera diferente para aprovechar los contenedores, un proceso comúnmente conocido como contenerización.

5.8.2.1 Diferencias con máquinas virtuales

Los contenedores a menudo se comparan con máquinas virtuales (VM). Al igual que las máquinas virtuales, los contenedores le permiten empaquetar su aplicación junto con bibliotecas y otras dependencias, proporcionando entornos aislados para ejecutar sus servicios de software.

Sin embargo, las similitudes terminan ahí, ya que los contenedores ofrecen una unidad mucho más liviana para que trabajen los desarrolladores y los equipos de operaciones de TI, lo que brinda una gran cantidad de beneficios.

- Los contenedores son mucho más ligeros que las máquinas virtuales.
- Los contenedores se virtualizan a nivel del sistema operativo, mientras que las máquinas virtuales se virtualizan a nivel de hardware.
- Los contenedores comparten el kernel del sistema operativo y usan una fracción de la memoria que requieren las máquinas virtuales.

5.8.3 Infraestructura como código (IaC)

Infraestructura como código (IaC) es la gestión y el aprovisionamiento de infraestructura a través de código en lugar de procesos manuales.

Con IaC, se crean archivos de configuración que contienen las especificaciones de su infraestructura, lo que facilita la edición y distribución de configuraciones. También garantiza que aprovisione el mismo entorno cada vez. Al codificar y documentar sus especificaciones de configuración, IaC ayuda a administrar la configuración y lo ayuda a evitar cambios de configuración ad hoc no documentados.

5.8.4 Markdown

Markdown[4] es un lenguaje de marcado ligero que se usa para agregar elementos de formato a documentos de texto sin formato. Creado por John Gruber en 2004, Markdown es ahora uno de los lenguajes de marcado más populares del mundo (Markdown Guide, s.f.).

[4] Markdown Guide, what is Markdown?

Markdown es diferente a usar un editor de texto como Microsoft Word, donde se hace clic en botones para dar formato a palabras y frases, y los cambios son visibles inmediatamente. Markdown, por lo contrario, crea un archivo en texto plano donde se agrega la sintaxis de Markdown al texto para indicar qué palabras y frases deben verse diferentes. Por ejemplo, para denotar un encabezado, se debe de agregar un signo de número antes del título (por ejemplo, **# Título Uno**). O para poner una frase en negrita, se agregan dos asteriscos antes y después de la frase (por ejemplo, ****este texto está en negrita****).

5.9 Base de conocimientos

Una base de conocimientos no es más que una librería o biblioteca de contenido que sirve para educar sobre una necesidad, un contexto, recursos, productos, servicios, procesos, o cualquiera otra índole de información. La consulta de la base de conocimientos generalmente está orientada a un enfoque de autoservicio, donde la persona o entidad interesada consulta de manera independiente la librería y obtiene el conocimiento que busca.

6. Objetivos

6.1 Objetivo general

1) Diseñar un plan de adaptación de prácticas DevOps para organizaciones, a alto nivel, incluyendo consideraciones prácticas de sus principios culturales, procesos y objetivos.

6.2 Objetivos específicos

1) Explorar y exponer casos de éxito en adaptaciones de DevOps destacando factores de éxito y aprendizajes más importantes.
2) Crear una base de conocimiento en formato digital de código libre relacionado a DevOps, de consumo autodidáctico o importado, para profesionales en el área de Tecnologías de Información o interesados.
3) Proponer prácticas de DevOps a empresas con procesos de ingeniería de software y profesionales de tecnologías de la información interesados.

4) Identificar las oportunidades de mejora en los procesos y flujos de trabajo de organizaciones que cuenten con procesos de ingeniería de software u operaciones de TI que utilicen métodos tradicionales de desarrollo, y ofrecer soluciones modernas basadas en los principios DevOps, para la optimización y mejora continua de los mismos.

7. Alcance

7.1 Sobre la guía de adaptación

La guía de adaptación de prácticas DevOps contendrá en el núcleo de su contenido etapas de evolución las cuáles describirán prácticas definitorias, así como aquellas que aportan al éxito de una adaptación apropiada. Dada la naturaleza de DevOps y su variable significado de éxito, la guía tendrá un enfoque descriptivo, exploratorio y sugestivo. El enfoque práctico se incluirá como recursos adicionales que la audiencia objetivo podrá utilizar. La guía presentará casos de estudio que demuestren el impacto que puede tener una adaptación exitosa.

La guía se presentará de manera educativo en formato digital y físico, con una identidad visual y editorial propia que sirva como factor de interés y atracción.

7.2 Sobre la base de conocimiento

La base de conocimiento de DevOps se presentará como un contenido de código abierto almacenado en un repositorio de un sistema de control de versiones público. El formato base de la base de conocimiento será documentado similar a un curso, para que pueda con facilidad ser utilizado para una modalidad impartida. De igual manera, se utilizará el formato Markdown para la documentación del contenido.

Para fines de demostración, se utilizará una plataforma de aprendizaje en línea para cargar el contenido del curso y confirmar su fácil importación a plataformas similares en caso de requerirlo.

7.3 Audiencia

7.3.1 Audiencia técnica ideal

DevOps es una disciplina que converge muchas aristas de la informática y de las tecnologías de información. Para tomar consumir los entregables de este trabajo de grado manera apropiada, se necesita un perfil técnico intermedio-avanzado para garantizar que el conocimiento es aprovechado al máximo.

Esto no implica que una persona con menos experiencia pueda consumir el contenido expuesto, solo expresa que existirán brechas de entendimiento o esfuerzos extras de aprendizaje para asimilar correctamente la intención del contenido. El rol del personal técnico también tiene incidencia, aunque en menor impacto. Se recomienda que se tenga previa experiencia en roles similares a:

- Administrador de Sistemas
- Ingeniero de Software
- Gestor de proyectos de TI

8. Marco metodológico

A nivel general, se plantea realizar un guía de adaptación y construir una base de conocimiento, lo que nos exige separar los entregables y las estrategias de trabajo en dos partes independientes, aunque relacionadas.

La guía de adaptación tiene un carácter teórico, explicativo y demostrativo. En contraste, la base de conocimiento está enfocada en el desarrollo profesional de individuos con conocimiento técnico intermedio capaces de entender los nuevos paradigmas que se presentan con las prácticas DevOps, por lo que es tanto teórica como práctico, Sin embargo, para ambos casos, el factor común es la construcción de contenido y su posterior presentación en los formatos descritos en el alcance.

8.1 Investigación y recolección de datos

Sin duda alguna DevOps es un movimiento emergente, pero existe bastante contenido a digerir de diferentes fuentes para solidificar los principios y valores que le caracterizan. Todo lo sustentado en este proyecto de grado plantea ser relativamente nuevo en el

contexto académico, pero no dejará de lado referencias y fuentes de valor de lo ya existente, que sea relevante al alcance ya definido.

8.1.1 Fuentes primarias

La fuente primaria de contenido utilizada fue la del programa de investigación denominado DORA State of DevOps[5]. El programa de investigación "representa siete años de investigación y datos de más de 32,000 profesionales en todo el mundo".

"Es la investigación de investigación académicamente rigurosa más antigua de su tipo, que proporciona una visión independiente de las prácticas y capacidades que impulsan el alto rendimiento en la entrega de tecnología y, en última instancia, los resultados organizacionales" (DevOps Research and Assessment LLC, n.d.).

Debido a que los datos recopilados por DORA representan en su mayoría datos verídicos de una gran variedad de individuos u organizaciones, este trabajo de grado se apoya de su basta biblioteca de conocimiento para sustentar en gran medida la investigación documentada, utilizando principalmente sus reportes de estado anuales y catálogo de capacidades.

8.1.2 Fuentes secundarias

De manera adicional, información valiosa y de utilidad académica se obtuvo desde las siguientes fuentes:

- Libros relacionados a optimización de flujos de desarrollo, modernización en procesos y tecnologías de desarrollo de software y operaciones, casos de estudio sobre ejecución de proyectos con objetivos de aumentar la resiliencia, seguridad y escalabilidad de sistemas.
- Artículos digitales, experiencias y anécdotas de equipos de trabajo en organizaciones de cualquier nivel a nivel de artículos web, noticias relacionadas, entre otros recursos encontrados en la Internet.
- Individuos capacitados en el ámbito tecnológico que implementan o participan en cualquier capacidad flujos de desarrollo de software y operaciones de sistemas.

[5] DORA's State of DevOps Research Program

Puede incluir Ingenieros, Gestores de proyectos, Gerentes de TI, especialistas y consultores, entre otros.

- Reportes estadísticos de la industria de TI, Ingeniería de Software y Seguridad de la información, en cuanto a tendencias, novedades, recomendaciones, entre otros contextos relevantes a los objetivos del proyecto.

8.1.3 Tipos de investigación

Se utilizó, en apoyo a los puntos anteriores, el tipo de investigación básica. Nuestro enfoque nos permite explorar más a fondo lo que representa el concepto de DevOps y proponemos prácticas que extienden el alcance del concepto a entidades que sean de interés al tema de estudio.

8.2 Organización de datos recolectados

Con el fin de realizar un análisis objetivo sobre la realidad de lo que se predica, existe y se espera de las prácticas DevOps, una vez recolectada toda la información necesaria desde las fuentes mencionadas se realizó un proceso de organización de datos para comparar las asunciones iniciales y tomar nota de aquellos valores que fueron de mayor interés para la construcción de la guía de adaptación y base de conocimiento ,y la forma en la que se presentan los entregables delimitados en nuestro alcance.

8.3 Creación de contenido

La creación de contenido empieza luego de la organización de los datos recolectados que a su vez empieza tras la recolección de estos. La creación de contenido se hizo en varias etapas iterativas y se utilizó una estrategia progresiva de refinamiento y mejora continua (basada en los principios de las metodologías ágiles).

El plan de ejecución general se basó en un contenido creado inicialmente de manera cruda, apoyándose de mapas mentales, notas asociadas y referencias. Se planteó un esqueleto con las ideas y conceptos iniciales, y construyendo progresivamente hasta lograr completar el contenido que cumpliría con los objetivos marcados.

De manera específica, las etapas de la creación de contenido utilizadas fueron:

8.3.1 Estructuración del esqueleto de contenido

Por esqueleto de contenido nos referimos principalmente a los distintos títulos y subtítulos que corresponden a la estructura del contenido. En sentido figurado, en esta etapa se definieron los pilares o tópicos principales a tratar para desde allí partir a la profundización y documentación del conocimiento que se desea exponer.

8.3.2 Descarga de contenido crudo

Como una iteración inicial, se descargó de varias fuentes confiables contenido que suplemente el esqueleto del contenido inicial. En esta etapa también se realiza una revisión rápida sobre la validez de la información que se coloca. En resumen, esta etapa se encarga de colocar contenido en el esqueleto definido en la etapa anterior sin profundizar ni refinar el contenido como se espera mostrar al final.

8.3.3 Refinamiento de contenido

En la etapa de refinamiento se toma la descarga cruda de contenido hecha en la etapa anterior y se normaliza tomando en cuenta la facilidad de lectura, simplicidad, y cobertura. Esta etapa puede iterarse en sí misma cuantas veces sea necesaria para alcanzar un grado de calidad apropiado.

8.3.4 Control de calidad

Una de las etapas finales consistió en proveer una mejor estructura y visualización del contenido previamente documentado y refinando, así como también correcciones gramaticales y de sintaxis.

8.4 Presentación de contenido

El contenido se preparó en diferentes formatos, con el fin de ser un contenido accesible para cualquier público con los conocimientos técnicos necesarios. Algunos de los formatos iniciales propuestos:

- Repositorios de código fuente, utilizando plataformas como GitHub[6], GitLab[7] o BitBucket[8].
- Maquetación de libro digital (e-book) con un proveedor editorial.
- Página web en formato Wiki[9] accesible desde cualquier navegador moderno.

Dependiendo de los apoyos visuales que se determinen durante la creación de contenido, los formatos finales pueden o no variar.

8.5 Modelo CAMS y escala evolutiva

Uno de los modelos más populares para la medición de madurez de adaptación de DevOps en organizaciones es el Modelo CAMS. La etimología de la palabra CAMS viene por las palabras en inglés **Culture**, **Automation**, **Measurement** y **Sharing** (Puppet & CircleCI, n.d.); que en español se traducen a **Cultura**, **Automatización**, **Medición** y **Compartir**. Estos cuatros pilares del modelo describen a alto nivel el éxito alcanzado de DevOps, y se utilizará en este trabajo de grado para demostrar el alineamiento de organizaciones con altos índices de evolución a nivel mundial con el fin de justificar y demostrar los patrones de éxito alcanzables por cualquier organización.

Para crear una escala evolutiva promedio, se obtuvo respuestas a cuatro preguntas, cada uno orientada a uno de los pilares del modelo CAMS:

Pilar	Pregunta	Posibles respuestas
Cultura	¿Dónde dirías que estás en cuanto a cultura en tu adaptación de DevOps hasta ahora?	a) Tenemos un solo equipo que tiene una fuerte cultura DevOps. b) Tenemos varios equipos dentro de un departamento con una sólida cultura DevOps. c) Tenemos un solo departamento que tiene una fuerte cultura DevOps. d) Tenemos una sólida cultura de DevOps en varios departamentos.
Automatización	¿Dónde diría que se encuentra en cuanto a automatización en su viaje de DevOps hasta ahora?	a) Los equipos están automatizando los servicios que controlan, para sus propias necesidades.

[6] github.com
[7] about.gitlab.com
[8] bitbucket.org
[9] Definicion de Wiki, Fundéu RAE

		b) Los equipos están automatizando los servicios que controlan, para las necesidades de otros. c) Los equipos están colaborando para automatizar los servicios para un uso amplio. d) Algunos servicios clave están disponibles a través del autoservicio. e) La mayoría de los servicios están disponibles a través de autoservicio.
Medición	¿Dónde diría que se encuentra en términos de medición en su viaje de DevOps hasta ahora? Seleccione todas las que correspondan.	a) Medimos manualmente las métricas clave del sistema (por ejemplo, el rendimiento de la computadora, el rendimiento, etc.). b) Medimos automáticamente las métricas clave del sistema. c) Las mediciones objetivas a nivel empresarial se recopilan manualmente utilizando métricas a nivel del sistema. d) Las mediciones objetivas a nivel empresarial están disponibles automáticamente bajo demanda.
Compartir	¿Dónde diría que está, en cuanto a compartir, en su viaje de DevOps hasta ahora?	a) Los patrones y las mejores prácticas se comparten dentro de los equipos. b) Los patrones y las mejores prácticas se comparten entre los equipos. c) Los patrones y las mejores prácticas se comparten en toda la organización. d) Los patrones y prácticas se comparten fuera de la organización.

Fuente: Puppet State of DevOps 2018

Tabla 1 Preguntas para encuesta de escala evolutiva del modelo CAMS

Para comprender el progreso de una organización para cada uno de los pilares de CAMS, "se utilizó un modelo para medir la posición de cada organización en una escala evolutiva. La escala evolutiva califica las respuestas en función de la frecuencia con la que la organización de los encuestados realizaba cada práctica (1 = Nunca, 2 = Rara vez, 3 = A veces, 4 = La mayor parte del tiempo, 5 = Siempre)" (Puppet). Luego se suman estos puntajes para crear un puntaje compuesto. Con base a este puntaje compuesto, luego

se agrupan las respuestas en tres categorías: **Bajo**, **Medio** y **Alto**. Las organizaciones que están empleando todas las prácticas con una alta frecuencia son altamente evolucionadas o Altas. Aquellas organizaciones que emplean prácticas con baja frecuencia son Bajas, y aquellas que realizan algunas prácticas a veces son Medianas.

8.6 Datos estadísticos y observaciones

Los datos estadísticos presentados en esta sección intentan justificar la importancia, el impacto y los resultados que se obtienen a través del alineamiento con los principios y prácticas de DevOps tomando en cuenta un contexto mundial. El alcance de estos datos no es específico de República Dominicana o de la región de LATAM, más bien, incluye todas las regiones de alto índice tecnológico (y desde donde se reportan los apremiados élite en el viaje evolutivo, incluyendo profesionales de todas partes del mundo.

Principalmente, los datos aquí recolectados y presentados forman parte de los Reportes de Estado de DevOps (en inglés, "State of DevOps reports") anuales, avalados por el proyecto de investigación DORA, y grandes entidades pioneras en el mundo de DevOps como son Puppet[10] y Google. Para 2021, se contaba con siete años de investigación y más de 32,000 respuestas a encuestas de profesionales de la industria.

8.6.1 Métricas de rendimiento de entrega

De acuerdo con el reporte Puppet State of DevOps 2019 "desde hace 8 años, los equipos de DevOps altamente evolucionados han demostrado constantemente un mejor rendimiento en cuatro métricas de rendimiento de software clave: **frecuencia de despliegues**, **tiempo de espera para cambios** y **tiempo de recuperación**, y **tasa de fallos por cambios**" (Puppet, s.f.).

[10] puppet.com

	Low	Mid	High
Deployment frequency	Monthly or less often	Between daily and weekly	On demands (whenever we want)
Lead time for changes	Between a week and 6 months	Less than a week	Less than an hour
MTTR	Less than a week	Less than a day	Less than an hour
Change failure rate	Less than 15%	Less than 15%	Less than 5%

Fuente: State of DevOps Report, 2018, Puppet Labs.

Figura 8-1 | Accelerate State of DevOps 2021, Software delivery and operational performance

Estas métricas también son validades por el Accelerate State of DevOps 2021, y las describen como "métricas de rendimiento de entrega" en vez de "métricas clave de rendimiento de software" como lo hace el Reporte Puppet. En el caso del reporte de Accelerate, se incluye una quinta métrica: "fiabilidad".

"La quinta métrica representa el desempeño operativo y es una medida de las prácticas operativas modernas. La métrica principal para el desempeño operativo es la confiabilidad, que es el grado en que un equipo puede cumplir sus promesas y afirmaciones sobre el software que opera. Históricamente, hemos medido la disponibilidad en lugar de la confiabilidad, pero debido a que la disponibilidad es un enfoque específico de la ingeniería de confiabilidad, hemos ampliado nuestra medida a la confiabilidad para que la disponibilidad, la latencia, el rendimiento y la escalabilidad estén representados de manera más amplia. Específicamente, les pedimos a los encuestados que calificaran su capacidad para cumplir o superar sus objetivos de confiabilidad. Descubrimos que los equipos con diversos grados de desempeño en la entrega obtienen mejores resultados cuando también priorizan el desempeño operativo" (Google, s.f.).

Debido a esto, estas cinco (5) métricas son las que determinan en gran medida una alta evolución organizacional en base a la adaptación de prácticas DevOps.

8.6.2 La nube

De acuerdo con Accelerate State of DevOps 2019, "un número cada vez mayor de organizaciones eligen soluciones de nube híbrida y multinube" (Google, s.f.).

En las encuestas realizadas en el Accelerate State of DevOps 2021, se preguntó a los encuestados dónde estaba alojado su servicio o aplicación principal, y el uso de la nube pública está en aumento. El 56% de los encuestados indicaron usar una nube pública (incluidas varias nubes públicas), un aumento del 5% con respecto a 2019 (Google, s.f.).

También se preguntó específicamente sobre el uso de múltiples nubes, y el 21% de los encuestados informó que se implementó en múltiples nubes públicas. El 21% de los encuestados indicó que no usaba la nube y, en cambio, usaba un centro de datos o una solución local. Finalmente, el 34% de los encuestados informa que usa una nube híbrida y el 29 % informa que usa una nube privada.

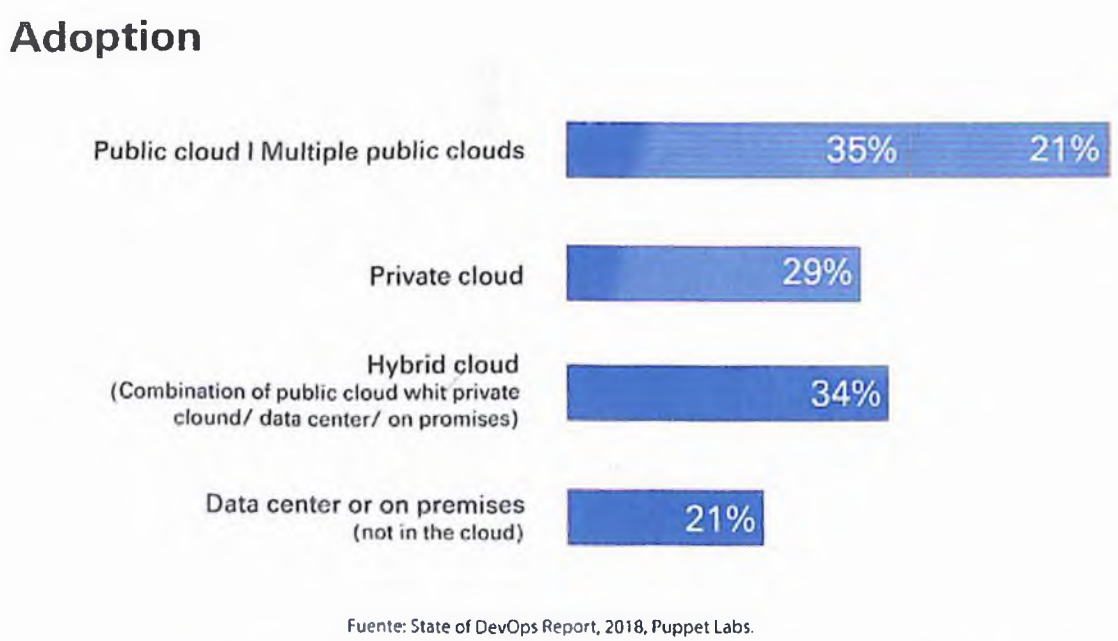
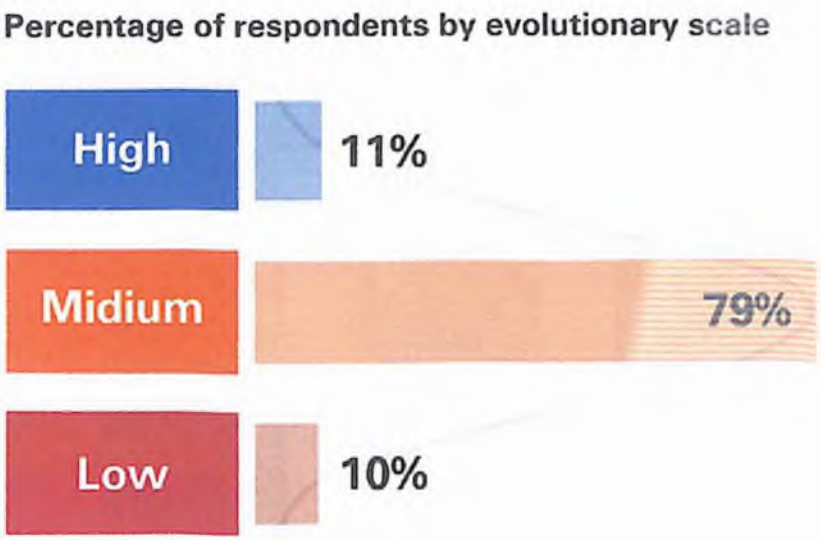


Figura 8-2 | Accelerate State of DevOps 2021, Software delivery and operational performance, Cloud Adoption

8.6.3 Porcentajes de niveles evolución sobre el modelo CAMS

El *90%* de las organizaciones encuestadas poseen una evolución **Media**. Casi el *11%* es de una evolución **Baja** y poco menos del *10%* es de **Alta** evolución. Esto nos dice que las prácticas de DevOps se han generalizado y que es mucho más difícil pasar de Medio a Alto que de Bajo a Medio.



Fuente: State of DevOps Report, 2018, Puppet Labs.

Figura 8-3 | Puppet State of DevOps 2018, CAMS and the evolutionary scale

8.6.4 Progreso cultural por escala evolutiva

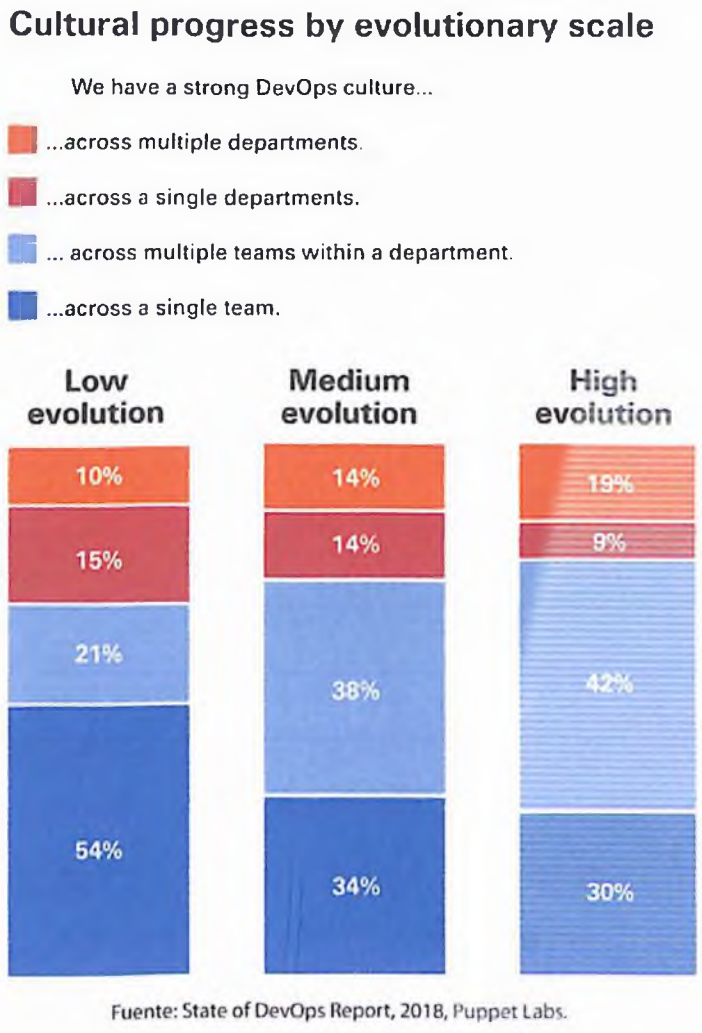
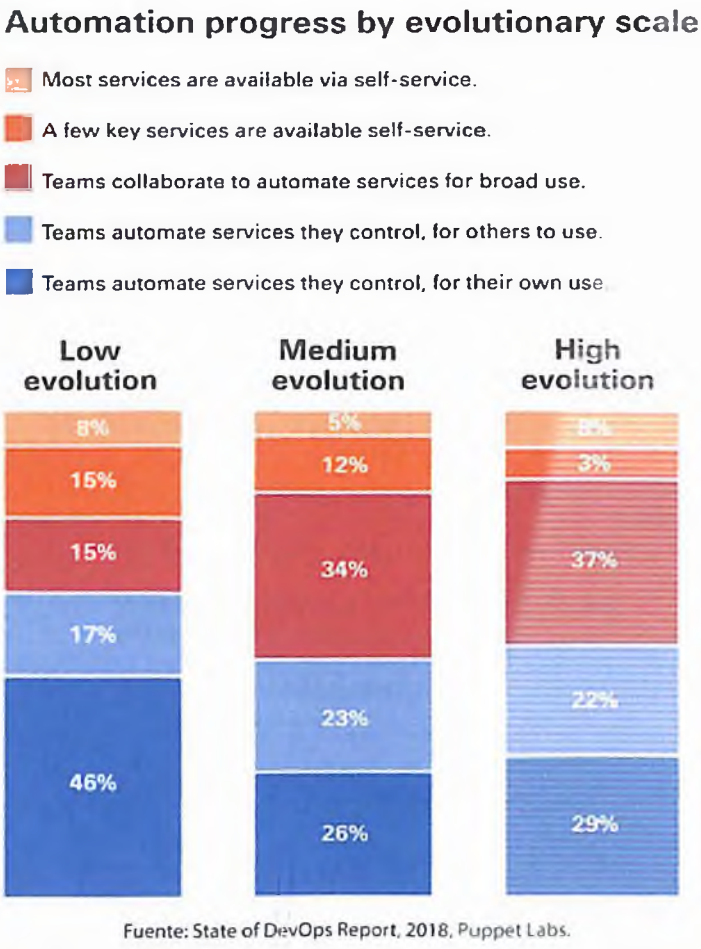


Figura 8-4 | Puppet State of DevOps 2018, CAMS and the evolutionary scale

Las organizaciones altamente evolucionadas tuvieron la menor cantidad de respuestas a *"Tenemos un solo equipo que tiene una fuerte cultura de DevOps"* y la mayor cantidad de respuestas a *"Tenemos varios equipos dentro de un departamento que tienen una fuerte cultura de DevOps"* (Puppet).

8.6.5 Progreso de la automatización por escala evolutiva



Fuente: State of DevOps Report, 2018, Puppet Labs.

Figura 8-5 | Puppet State of DevOps 2018, CAMS and the evolutionary scale

Vemos una clara mejora de las organizaciones Bajas a las Medias y Altas cuando se trata de equipos que colaboran para automatizar servicios que se consumen ampliamente. También vemos una caída correspondiente en la proporción relativa de servicios que están automatizados solo para el consumo interno del equipo.

"Nuestra hipótesis es que la diferencia mínima en el grado de automatización entre las cohortes Media y Alta refleja el hecho de que la automatización es posiblemente el pilar más fácil de implementar del modelo CAMS. El personal técnico comprende bien la automatización, tiene un camino relativamente predecible y puede tener éxito si brinda a los profesionales técnicos disciplinados el tiempo y el ancho de banda para automatizar, junto con un mandato para hacerlo" (Puppet).

Sin embargo, DevOps no se trata solo de **automatización**.

Los cambios culturales que se requieren para el éxito de DevOps son significativamente más difíciles de implementar y requieren aportes y apoyo organizacionales más amplios. También es más difícil medir los resultados en términos que la empresa pueda entender. Y donde es bastante fácil encontrar ejemplos destacados de automatización que pueda emular, es más difícil transferir la experiencia de evolución cultural de una organización directamente a otra organización y obtener un resultado exitoso.

8.6.6 Medición por escala evolutiva

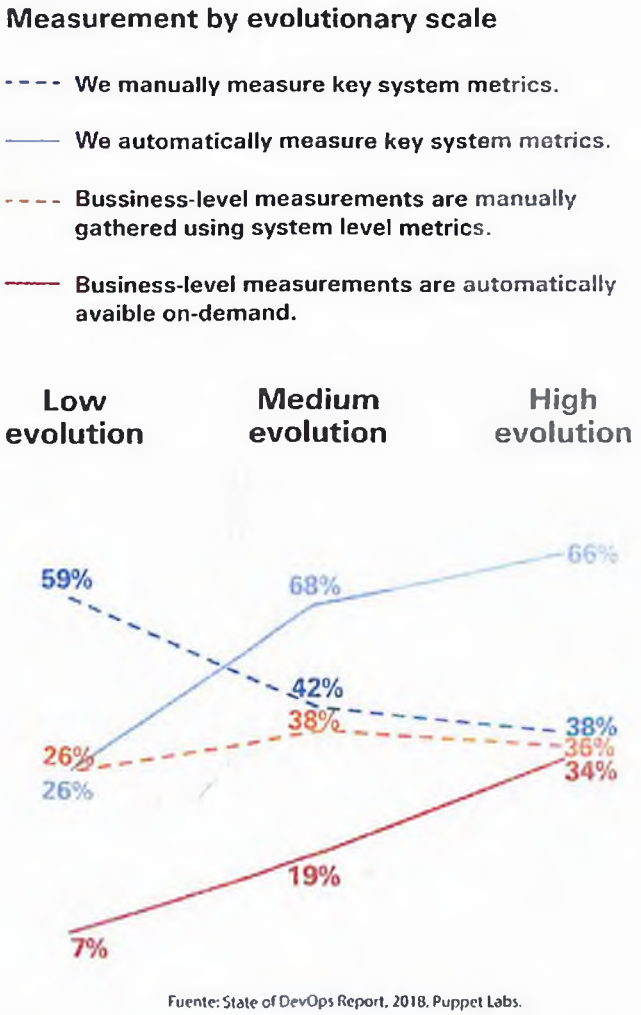


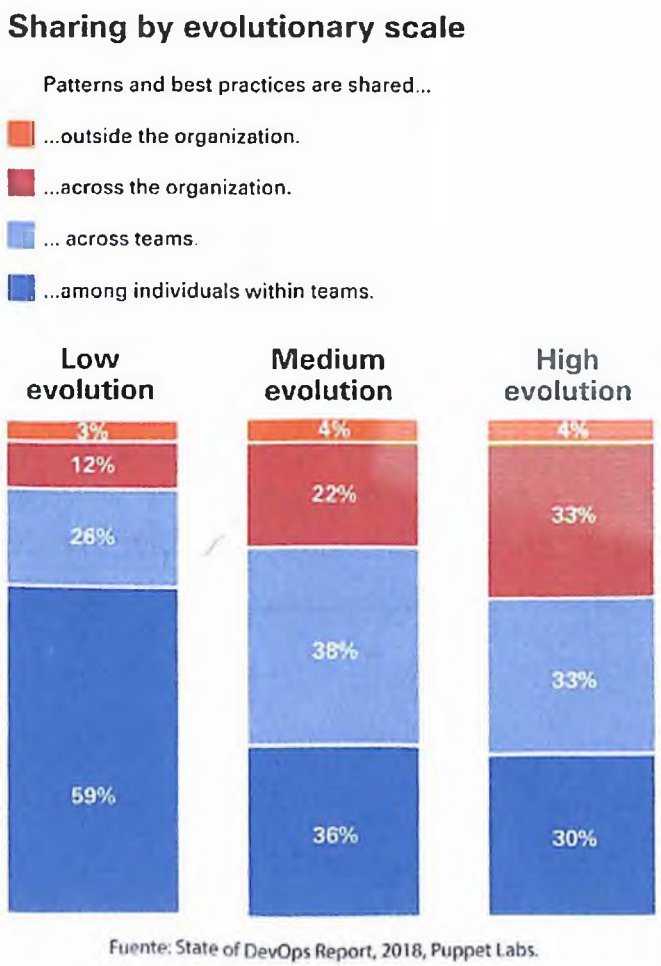
Figura 8-6 | Puppet State of DevOps 2018, CAMS and the evolutionary scale

Al observar estos datos, podemos crear la hipótesis para la medición de que una organización DevOps más madura tendría más probabilidades de haber automatizado una colección de métricas clave, y que esas métricas se referirían principalmente a los

objetivos comerciales. "Las organizaciones altamente evolucionadas tienen niveles significativamente más altos de métricas comerciales automatizadas que están disponibles bajo demanda, así como el nivel más alto de medición de sistemas automatizados. Las organizaciones altas también tienen el nivel más bajo de métricas del sistema recopiladas manualmente" (Puppet).

Los tres grupos (**alto**, **medio** y **bajo**) tenían aproximadamente la misma proporción de métricas comerciales recopiladas manualmente. Sospechamos que esto se debe al hecho de que normalmente hay un conjunto estándar de métricas comerciales que las personas están acostumbradas a recopilar manualmente, ya sea que estén haciendo DevOps o no, como ingresos, tasas de renovación, costos de adquisición de clientes, costos generales y costos variables. porcentajes.

8.6.7 Compartir por escala evolutiva



Fuente: State of DevOps Report, 2018, Puppet Labs.

Figura 8-7 | Puppet State of DevOps 2018, CAMS and the evolutionary scale

A medida que avanza la evolución de DevOps, vemos que aumenta el intercambio de información dentro de la organización, lo cual es un resultado esperado. Las organizaciones altamente evolucionadas pasan de compartir principalmente dentro de los equipos individuales a compartir ampliamente en toda la organización. "Hemos descubierto que este intercambio de mejores prácticas y patrones tiende a ir de la mano con mayores grados de automatización bien diseñada" (Puppet).

Tiene sentido: a medida que los sistemas de automatización evolucionan y consisten en abstracciones de alto nivel cada vez más integradas con el resto de sus sistemas, se vuelve cada vez más fácil compartir patrones de implementación y automatización. La implementación y el consumo de estos sistemas de automatización abstraídos se vuelven más simples y rápidos.

8.7 Estadísticas de mercado

Según la firma de investigación Allied Market Research, "...el tamaño del mercado global de DevOps se valoró en 6,780 millones de dólares en 2020 y se prevé que alcance los 57,900 millones de dólares en 2030, registrando una tasa compuesta de crecimiento anual (CAGR[11]) del 24,2% de 2021 a 2030".

Otro estudio más reciente, de parte de GlobeNewswire en el 2021, concluyó que "El tamaño del mercado global DevOps representó USD 7,398 millones en 2021 y se espera que alcance los USD 37,227 millones para 2030 con una CAGR considerable del 20% durante el período de pronóstico de 2022 a 2030" (GlobalNewswire, 2021). Este último estudio se observa un poco más conservador respecto al primero, pero demuestra de todos modos que el crecimiento de DevOps aún no ha alcanzado su cúspide, sobre todo asumiendo que estos porcentajes tienen mayor incidencia en las regiones de Norte América y Europa.

8.7.1 Impacto de la pandemia de COVID-19

La firma Allied Market Research comenta en el mismo reporte de investigación que "El brote de COVID-19 tuvo un impacto positivo en el crecimiento del mercado de DevOps debido al aumento repentino de la demanda de software y aplicaciones en línea

[11] Compound Annual Growth Rate. What you should know

basadas en la web entre las empresas y la creciente necesidad de una solución DevOps entre las empresas para agilizar el proceso de desarrollo de aplicaciones de software mediante la promoción colaboración entre varias entidades del ciclo de vida del desarrollo de software" (Allied Market Research, n.d.). Esto deja en evidencia que DevOps fue una herramienta de gran utilidad para las organizaciones que desarrollaron software durante la pandemia, cuyo alcance fue global.

8.8 Diseño de contenido

8.8.1 Estructura de la guía de adaptación

La guía de adaptación fue diseñada con el propósito de ser presentada en formato digital e impreso editorial en formato manual. La estructura de la guía partió con el esquema de presentar la información lo más directa y simple posible, tomando en cuenta la dimensión del contenido y de la audiencia esperada.

Las secciones principales de la guía de adaptación son:

#	**Sección**	**Descripción**
00	Resumen ejecutivo	El resumen ejecutivo servirá como introducción a la guía de adaptación, detallando de qué tratará invitando al lector a motivarse a leer el contenido.
01	Audiencia objetivo	Plantea delimitar cuál sería la audiencia principal de la guía, con el fin de poder maximizar la transferencia de información. DevOps exige cierto conocimiento previo sobre tecnologías de información por lo tanto esta sección informará al lector sobre la posible barrera de conocimientos con la que se puede encontrar.
02	DevOps: Introducción	Orienta sobre el tópico principal de la guía para nivelar el entendimiento previo versus el propuesto por la guía. El lector conocerá o confirmará su conocimiento sobre DevOps antes de entrar en materia con el contenido más técnico y complejo.
03	Modelo CAMS	Existen varios modelos o marcos de referencia sobre lo que se suponer debe ser DevOps. El modelo CAMS es el elegido para esta guía, delimitando los principios

		culturales en los cuales nos enfocaremos. Esta sección educa al lector sobre el modelo,
04	Etapas de evolución de DevOps	Las etapas de evolución corresponden al contenido práctico de la guía. Orienta al lector sobre cómo lograr iniciar el viaje evolutivo y categoriza las etapas acordes a prácticas definitorias y de éxito.
05	Cultura: Cómo transformar	Dado que DevOps implica una transformación digital, se incluyen algunas pautas y mejores prácticas sobre cómo lograr una transformación exitosa.
06	Casos de estudio	Presentación de casos de éxito en implementaciones de prácticas DevOps. Se muestra un caso local, así como uno internacional.
07	Conclusiones	Concluye el contenido descrito con algunos comentarios finales sobre la importancia del contenido expuesto.

Tabla 2 | Guía de adaptación de DevOps, Estructura de contenido

8.8.2 Identidad gráfica de la guía de adaptación

La identidad gráfica de la guía de adaptación se creó tomando en cuenta aspectos variados de la psicología color. La intención del contenido tiene como propósito transmitir la sensación de innovación, mejora y simplicidad del aprendizaje. A su vez, tener una presentación moderna y atractiva para cualquier tipo de público.

8.8.2.1 Colores

Los colores principales elegidos son el Azul y el Naranja. Su elección y justificación corresponde directamente con la descripción de éstos detrás de los objetivos ya mencionados. Ampliando en la descripción de cada color:

Azul: A parte de ser un color preferido por muchas personas, se asocia con calma, confianza, seguridad; por esto "funciona muy bien para sitios informativos de tecnología [...]" (Studio, 2019).

Naranja: A fines de evitar que el lector tenga síntomas de cansancio y desista del contenido, el color naranja "es usado para atraer la atención y provocar alegría" (León, 2018)

Los colores utilizados y sus códigos hexadecimal y RBH se muestran en la siguiente tabla:

Color	Código Hexadecimal	Código RGB	Muestra
Azul	25408F	rgb(37, 64, 143)	
Naranja	F15A29	rgb(241, 90, 41)	

Tabla 3 | Guía de adaptación de DevOps, Identidad gráfica. Colores

8.8.3 Presentación visual de la guía de adaptación

La presentación visual completa de la guía de adaptación es accesible en:

- issuu.com/aa16-1088/docs/guia_de_adaptacion_de_practicas_devops

8.8.3.1 Vista – Portada

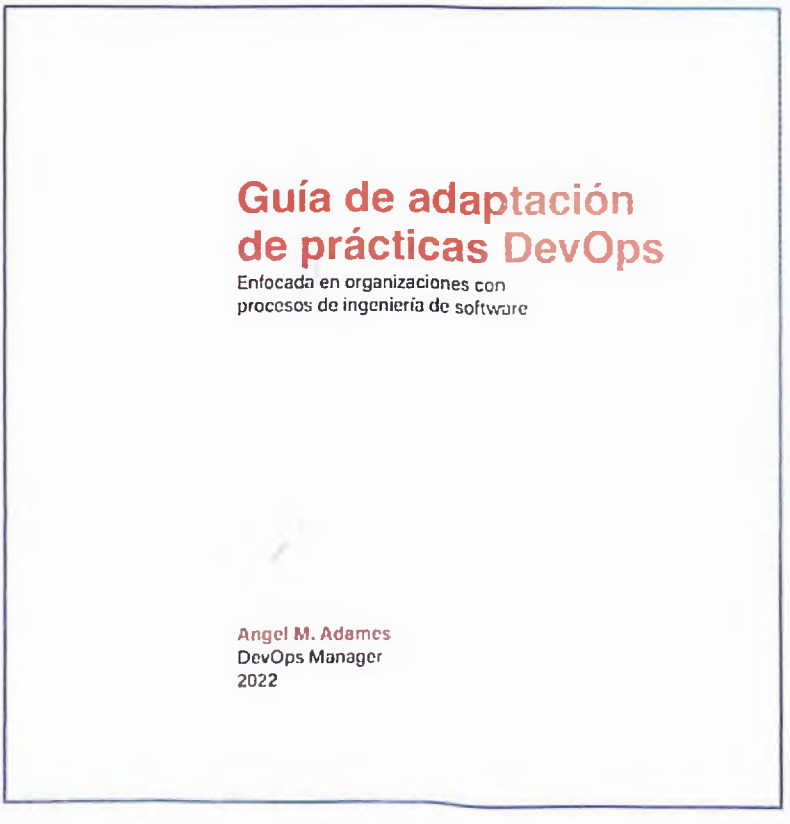


Ilustración 1 | Guía de adaptación de practicas DevOps. Portada

8.8.3.2 Vista – Tabla de contenidos

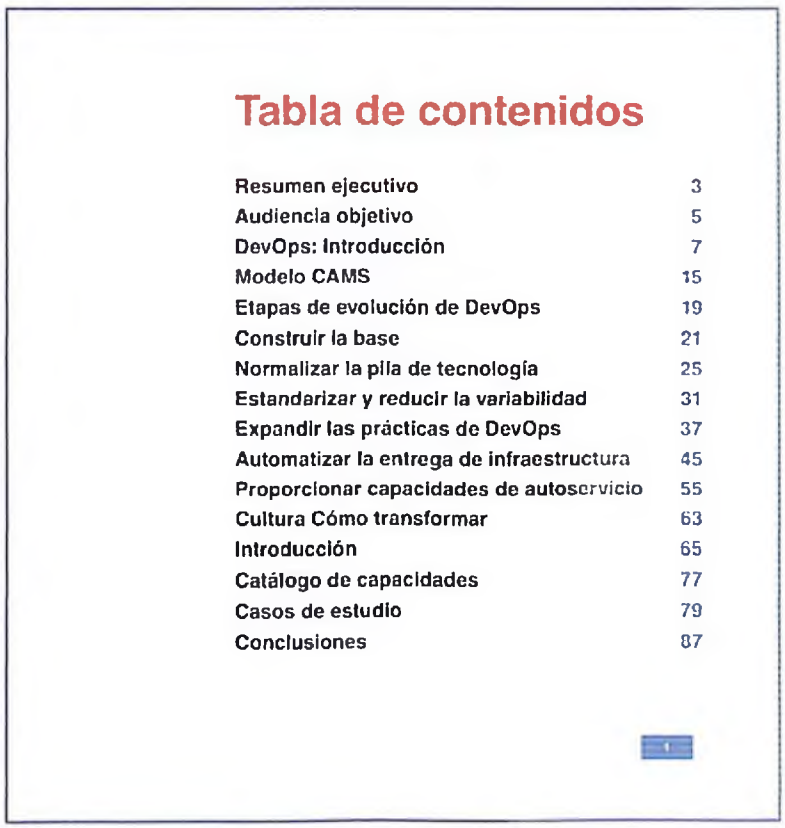


Tabla de contenidos

Ilustración 2 | Guía de adaptación de prácticas DevOps, Portada

8.8.3.1 Vistas – Sección 'DevOps: Introducción'

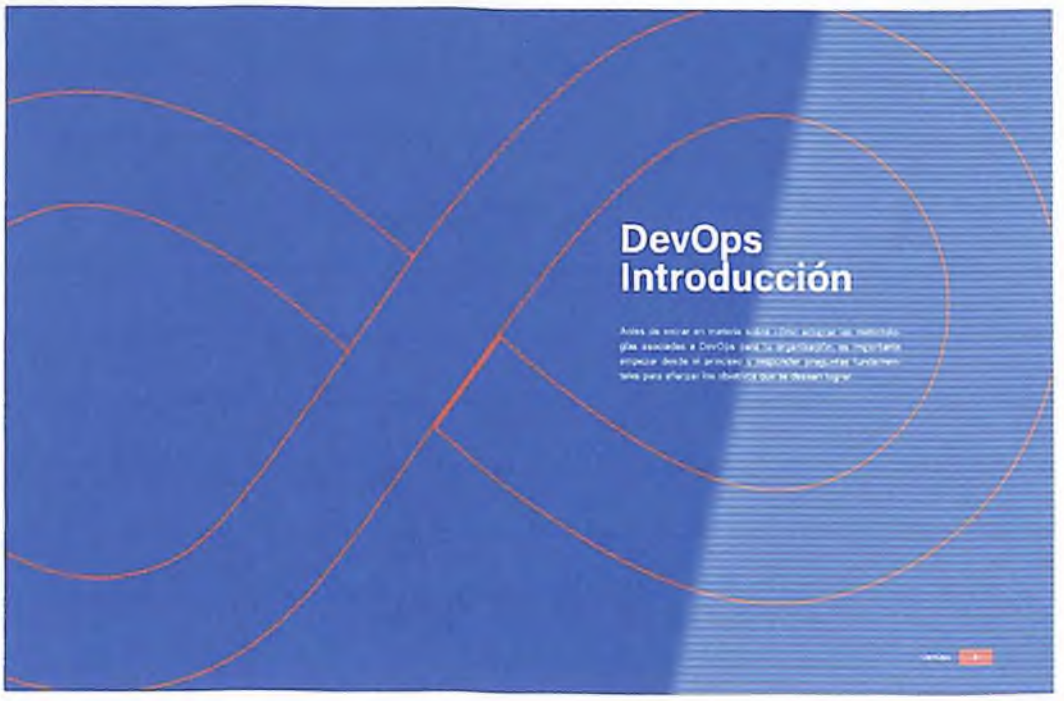


Ilustración 3 | Guía de adaptación de prácticas DevOps. Portada 'DevOps Introducción'

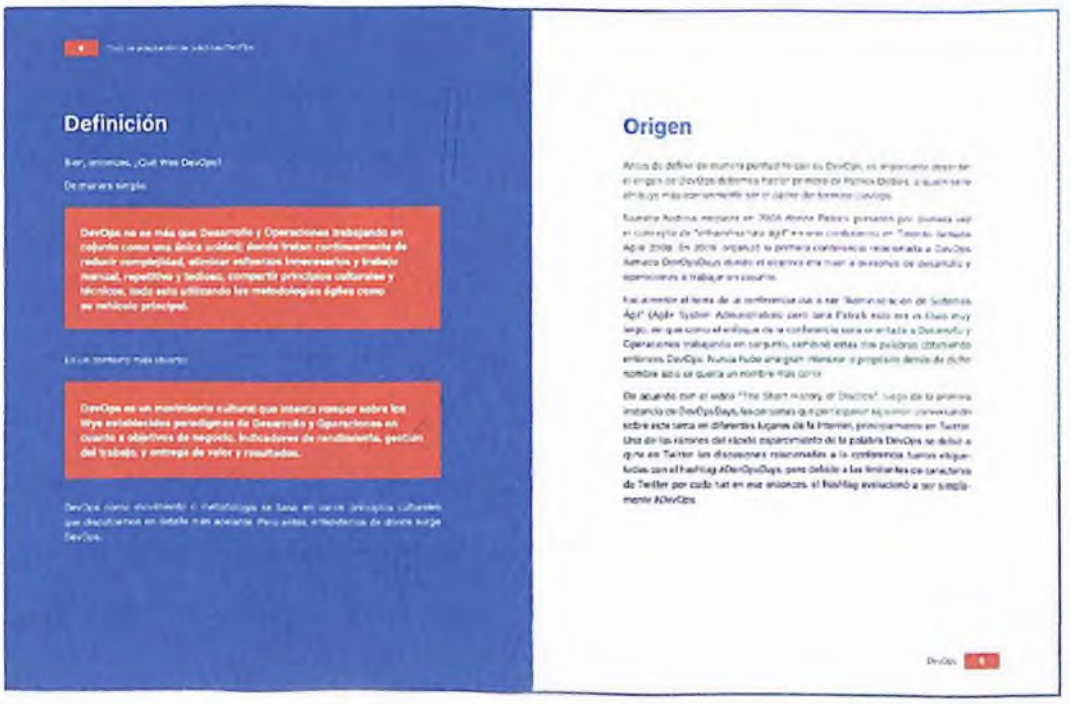
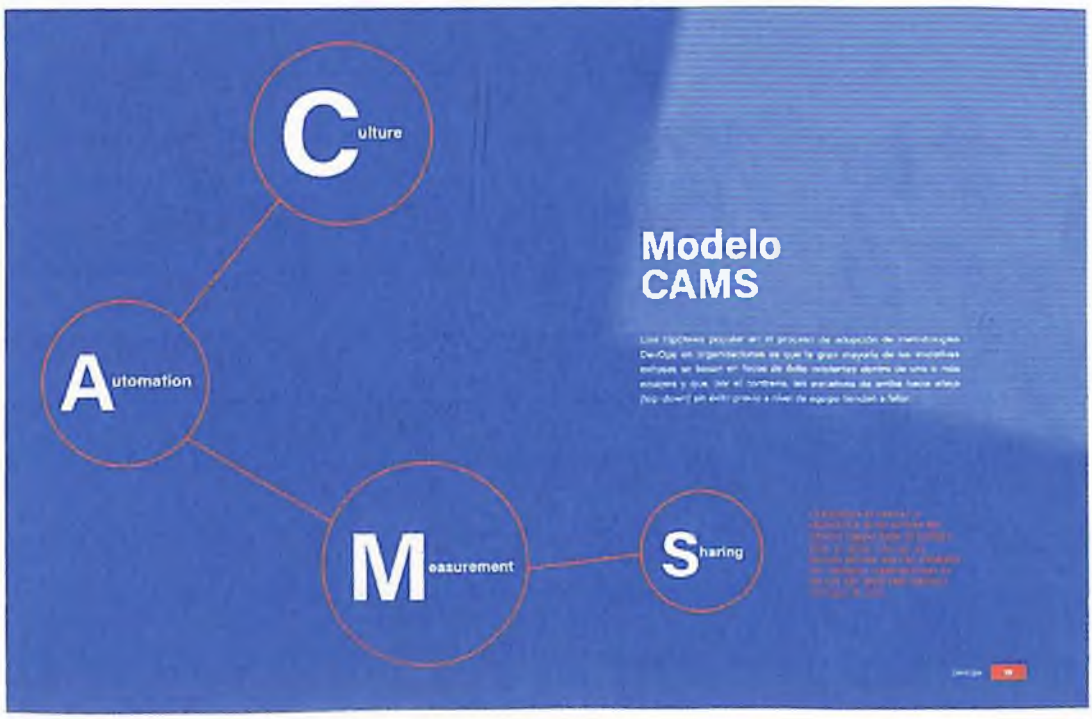


Ilustración 4 | Guía de adaptación de prácticas DevOps. Definición y Origen

Ilustración 5 | Guía de adaptación de prácticas DevOps, Principios culturales

8.8.3.2 Vistas – Sección 'Modelo CAMS'



Ilustracion 6 | Guía de adaptación de prácticas DevOps, Portada Modelo CAMS

8.8.3.3 Vistas – Sección 'Etapas de evolución'

Ilustración 7 | Guía de adaptación de prácticas de DevOps. Portada Etapas de evolución

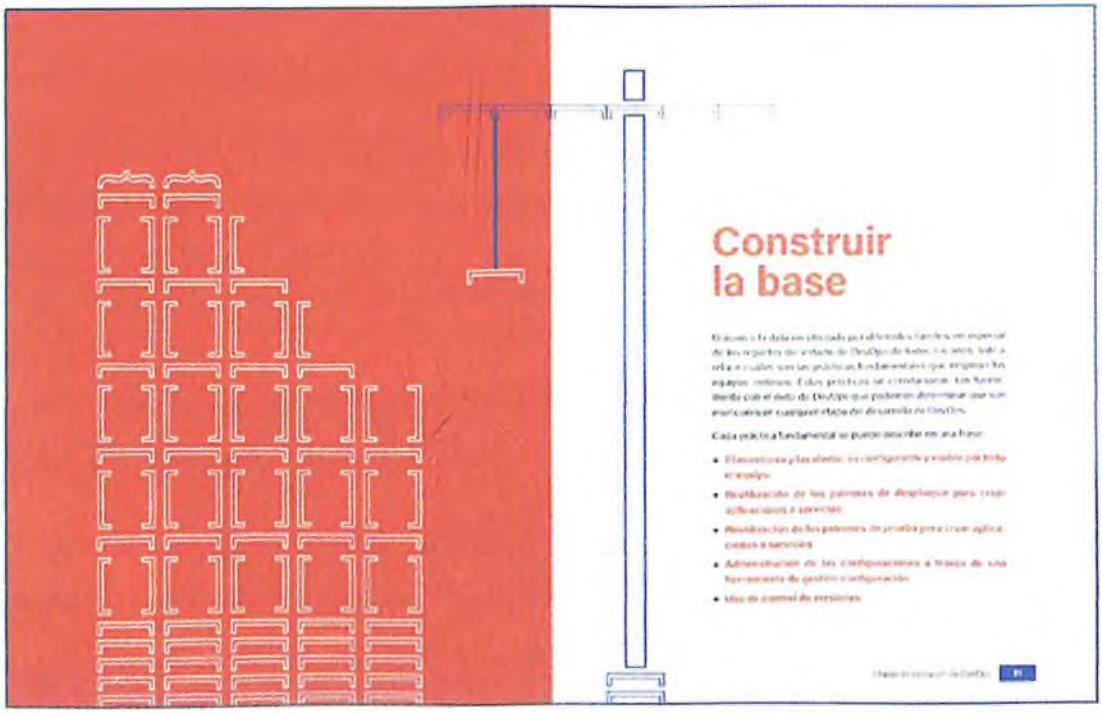


Ilustración 8 | Guía de adaptación de prácticas de DevOps, Construir la base

8.8.4 Estructura de la base de conocimiento

La estructura de la base de conocimiento es similar a la de curso académico virtual, pues consta de tres componentes principales que se asemejan a las mejores prácticas didácticas que se utilizan en muchos contextos educativos. Los tres componentes principales de la base de conocimiento son:

- Módulos
- Temas
- Laboratorios, temas y exámenes de autoevaluación

8.8.4.1 Módulos

Los módulos son la agrupación de temas bajo un contexto de interés común. En sentido general, comprenden el contenido del curso clasificado de manera secuencial y por conceptos de alto nivel. Los módulos serán la agrupación de más alto nivel en cuanto al contenido de la base de conocimiento. El curso está compuesto por 10 módulos, ignorando el módulo 0, que es de introducción a la base de conocimiento o curso.

8.8.4.2 Temas

Cada módulo se dividirá en varios temas. Cada tema comprende conceptos específicos y se trabajará de manera detallada a través de:

- Teoría (descripción conceptual, importancia, beneficios, desventajas, mejores prácticas)
- Demostraciones prácticas, donde aplique
- Exposición de datos curiosos

De manera adicional, cada tema tendrá contenidos de interés relacionados que el estudiante podrá incursionar para extender o enriquecer el conocimiento adquirido.

8.8.4.3 Laboratorios, temas y exámenes de autoevaluación

Algunos temas tendrán asignaciones prácticas que realizar. Las asignaciones pueden ser laboratorios, tareas y/o exámenes de autoevaluación.

Los laboratorios serán escenarios técnicos controlados realizados de manera local en el computador de trabajo. Los laboratorios usualmente requieren aprovisionamiento, configuración y ejecución de diferentes tipos de recursos y servicios.

8.8.5 Programa de la base de conocimiento

El programa de contenido posee una estructura similar a la de un curso académico.

Índice	**Módulo**	**Temas**
00	Introducción al curso	- Introducción al curso - Audiencia objetivo
01	Fundamentos de DevOps	- Introducción a DevOps - Metodologías ágiles - Procesos, principios culturales y herramientas
02	Computación en la nube	- Servicios en la nube - Proveedores de la nube
03	Ciclo de desarrollo de software	- Ciclo de vida del desarrollo de software (SDLC) - Control de versiones - Gestión de paquetes y dependencias
04	Contenerización	- Contenedores - Docker, Docker Compose
05	Integración continua	- Pruebas de integración - Herramientas de integración continua (CI) - Mejores prácticas de integración continua (CI)
06	Entrega continua	- Despliegue continuo - Estrategias de despliegue y patrones de lanzamiento - Herramientas de entrega continua (CD) - Gestión de versiones
07	Infraestructura como código	- Administración de sistemas - Shell Scripting - Gestión de la configuración - Orquestación de contenedores - Aprovisionamiento de infraestructura - Introducción a Terraform
08	Monitoreo y alertas	- Monitoreo de infraestructura - Monitoreo de aplicaciones - Gestión de logs - Observabilidad
09	Aseguramiento de la calidad	- Pruebas de software - Pruebas automáticas - Pruebas de carga y rendimiento

10	Seguridad continua	- DevSecOps - Pruebas de seguridad

Tabla 4 | Base de conocimiento de DevOps, Programa

8.8.6 Presentación web de la base de conocimiento

La presentación web de la base de conocimiento es accesible en la dirección: https://devops-kb.netlify.app.

8.8.6.1 Vista – Página de inicio

La página de inicio de la base de conocimiento consta con elementos reducidos para facilitar la inducción a la misma. En primera instancia, vemos el título de la página seguido de una pequeña descripción y el botón enlazado de "¡Empezar!"; este nos llevaría a la biblioteca de conocimientos y todos los módulos, temas y laboratorios que allí están documentados.

Ilustración 9 | Base de conocimiento de DevOps, Página de inicio

8.8.6.2 Vista – Sección 'Antes de Empezar'

Todas las secciones de la base de conocimiento siguen una estructura similar a la vista en la Ilustración 11. A la derecha se encuentra una tabla de contenido que permite navegar los diferentes títulos. A la izquierda el índice de todos los módulos y sus contenidos. En

la parte superior un panel de búsqueda, un ícono de enlace a GitHub y otro para habilitar el tema negro del sitio.

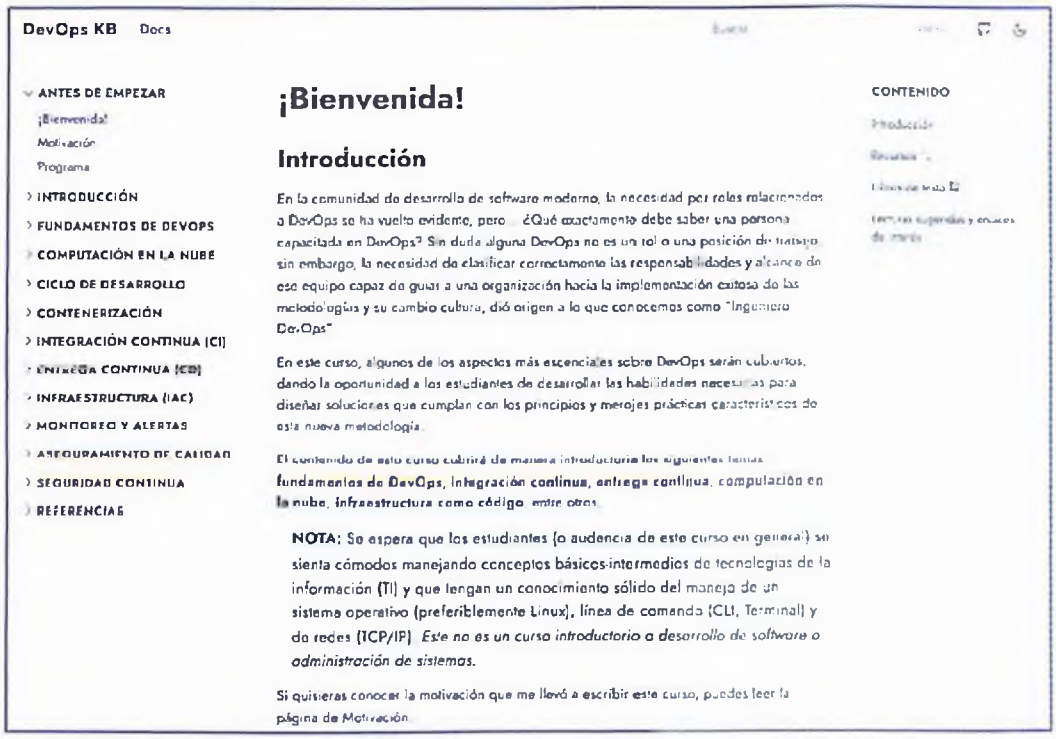


Ilustración 10 | Base de conocimiento de DevOps, Antes de empezar

8.8.6.3 Vista – Sección 'Introducción'

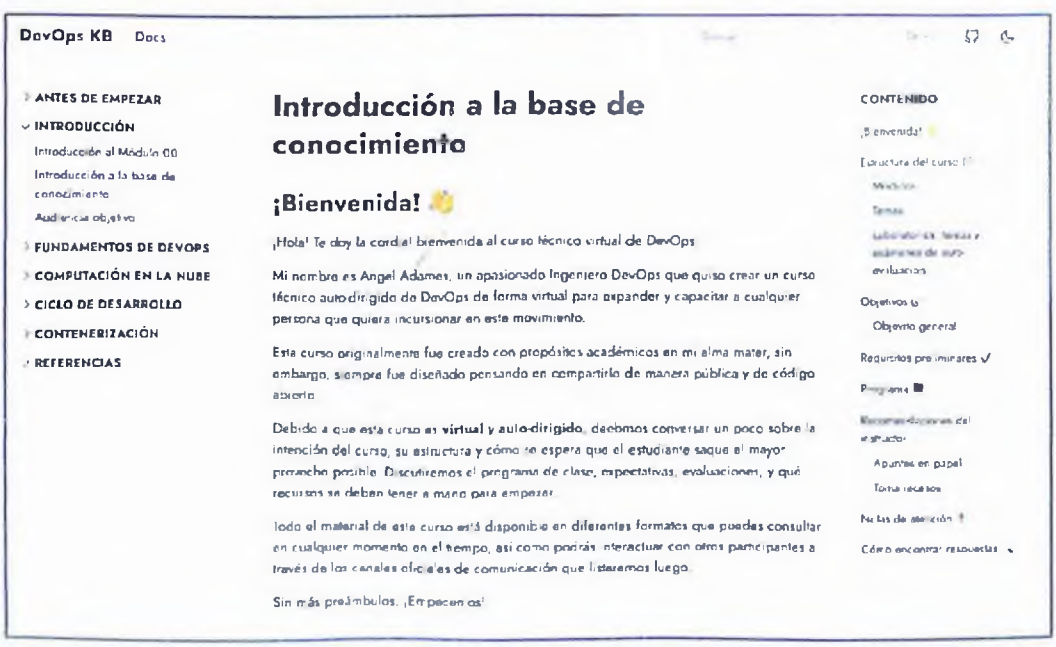


Ilustración 11 | Base de conocimiento de DevOps, Introducción

8.8.6.4 Vista – Sección 'Fundamentos de DevOps'

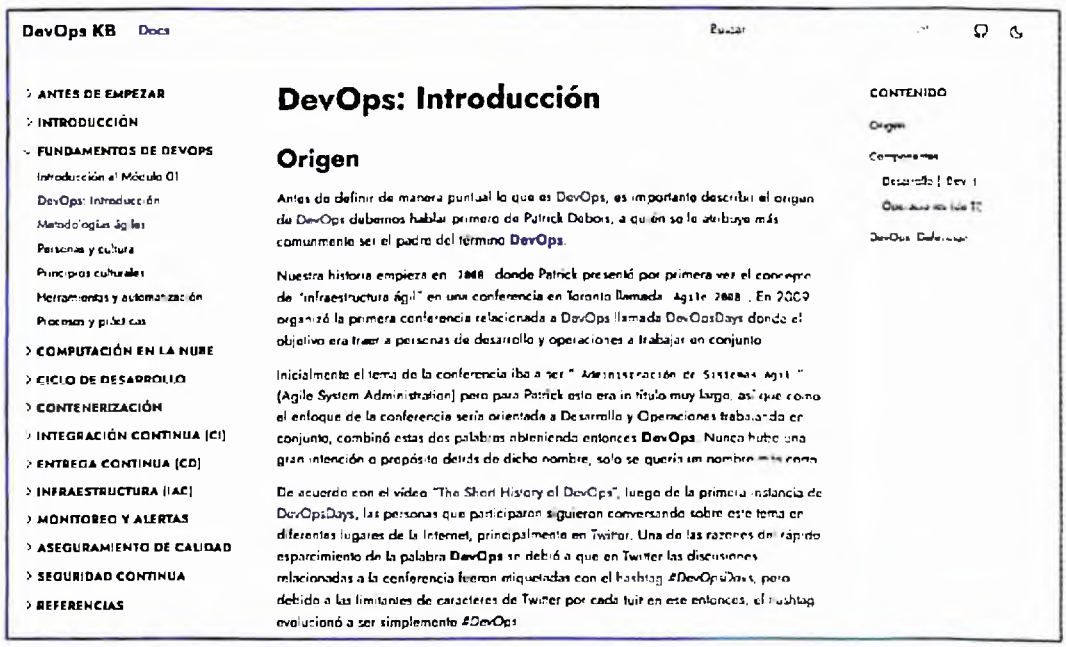


Ilustración 12 : Base de conocimiento de DevOps, DevOps: Introduccion

8.8.6.5 Vista – Sección 'Computación en la nube'

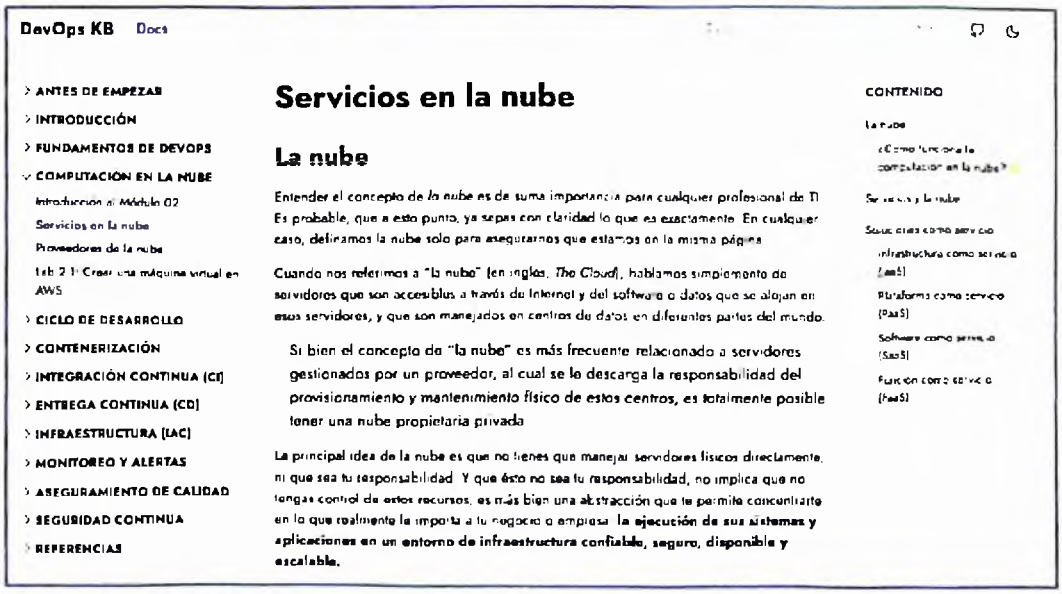


Ilustración 13 : Base de conocimiento de DevOps, Servicios en la nube

8.8.6.6 Vista – Sección 'Ciclo de Desarrollo'

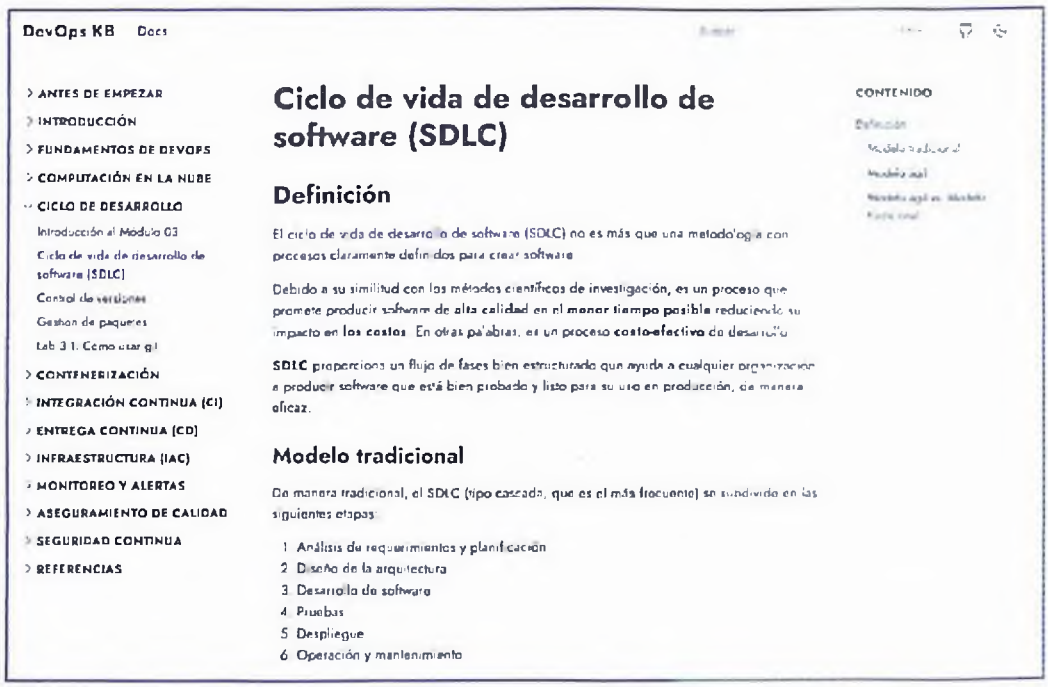


Ilustración 14 | Base de conocimiento de DevOps, Ciclo de vida de desarrollo de software (SDLC)

8.8.6.7 Vista – Sección 'Contenerización'

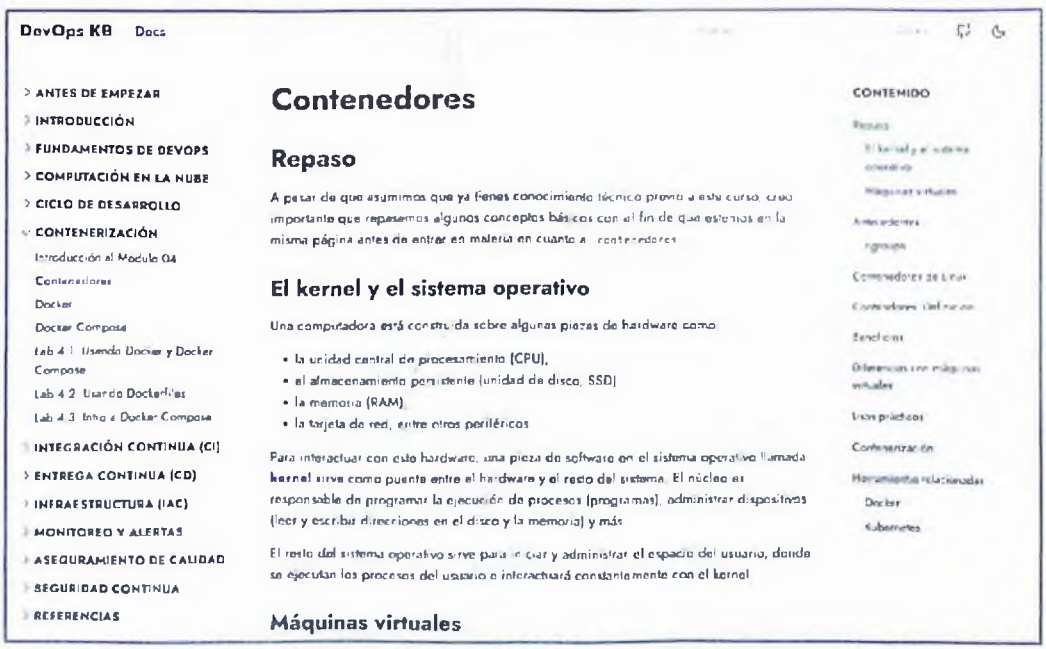


Ilustración 15 | Base de conocimiento de DevOps, Contenedores

8.8.6.8 Vista – Sección 'Integración continua'

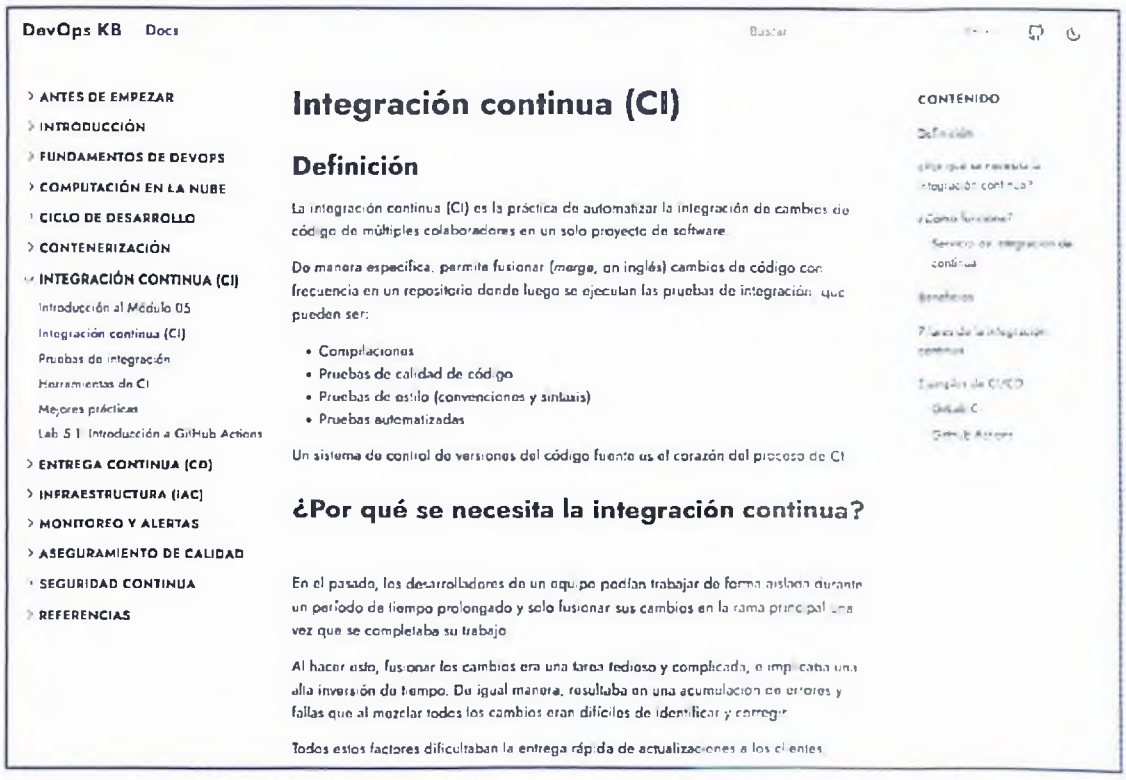


Ilustración 16 | Base de conocimiento de DevOps, Integración continua (CI)

8.8.6.9 Vista – Sección 'Entrega continua'

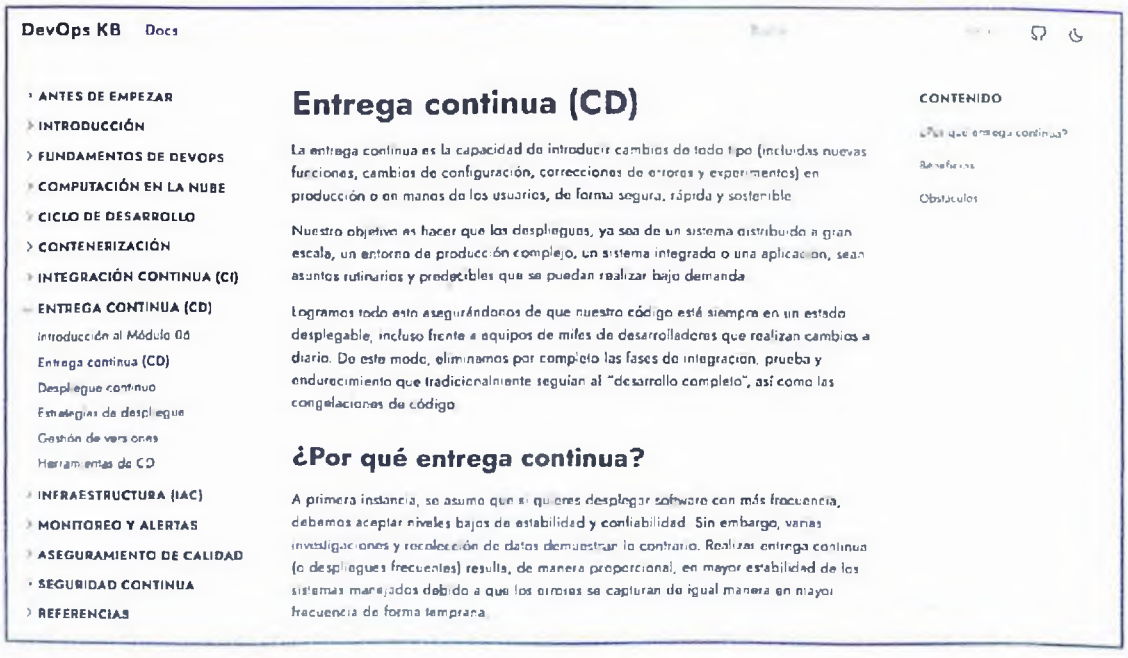


Ilustración 17 | Base de conocimiento de DevOps, Entrega continua (CD)

8.8.6.10 Vista – Sección 'Infraestructura'

Ilustración 18 | Base de conocimiento de DevOps, Introducción a Terraform

8.8.6.11 Vista – Sección 'Monitoreo y alertas'

Ilustración 19 | Base de conocimiento de DevOps, Monitoreo de infraestructura

8.8.6.12 Vista – Sección 'Aseguramiento de calidad'

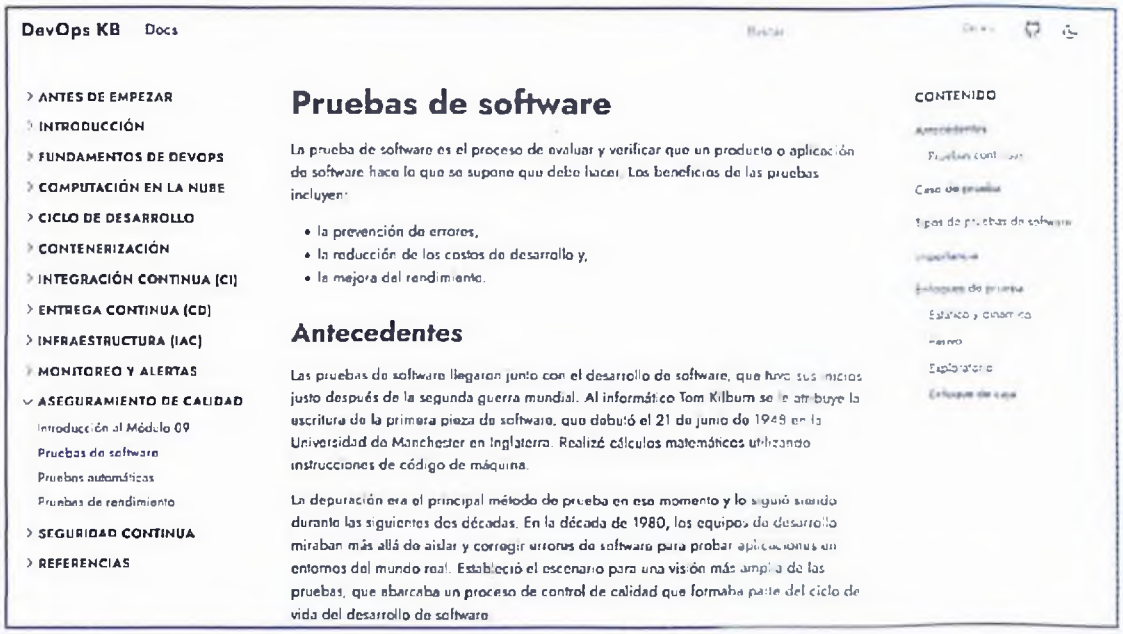


Ilustración 20 | Base de conocimiento de DevOps, Pruebas de software

8.8.6.13 Vista – Sección 'Seguridad continua'

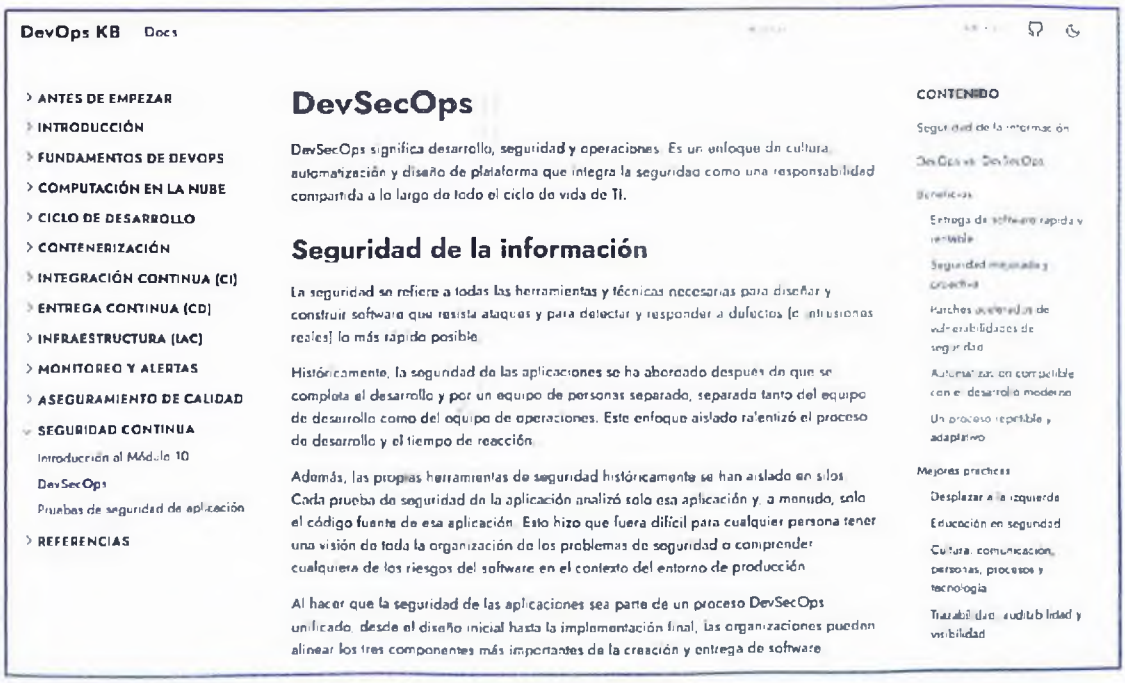


Ilustración 21 | Base de conocimiento de DevOps, DevSecOps

9. Factibilidad

9.1 Factibilidad operativa

Tanto la guía de implementación como la base de conocimiento tienen aspectos de creación o construcción casi exclusivamente de documentación. La documentación, como parte elemental de los entregables de este proyecto final, es viable debido a la disponibilidad de información existente, la experiencia de los autores en la materia a tratar producto de años desempeñando funciones relacionadas, y el tiempo disponible.

Para generar un impacto positivo en base a la documentación y creación de contenido, se necesita una presentación adecuada para atraer personas interesadas a consumirlo. Principalmente sobre la guía de implementación, la presentación se hará a través de un manual o guía digital diseñado por una empresa de editorial para obtener los mejores resultados posibles. En el caso de la base de conocimiento, se expone tanto un repositorio en control de versiones de GitHub como en una página web para facilidad de consumo y navegación.

La factibilidad operativa de los entregables, de utilizarse apropiadamente, dependerá en gran medida de la entidad o individuos que le utilicen. El contenido que se documenta no es necesariamente una guía práctica con pasos específicos a realizar; es más bien un marco de referencia para dar los primeros pasos en la evolución hacia una cultura DevOps. Por tanto, de aquí se excluye un análisis exhaustivo de la factibilidad operacional de los resultados que se pudiesen obtener por la utilización de los entregables.

9.2 Factibilidad técnica

El autor principal del contenido a producir posee una trayectoria de 4 años de experiencia desempeñando un rol asociado a la implementación de prácticas DevOps para diferentes organizaciones y empresas locales e internacionales. Se establece así que el personal a cargo de este proyecto tiene las capacidades y competencias requeridas, en base a su experiencia, para trabajar los objetivos definidos.

9.3 Factibilidad de tiempo

El tiempo aproximado para la ejecución de los objetivos propuestos se extiende a lo largo de **10 semanas**. El desglose de actividades se detalla en el cronograma de actividades del proyecto.

9.3.1 Cronograma de actividades

Guía de adaptación de prácticas DevOps
Cronograma de actividades

#	Actividades	Estimación Horas	Estimación Días	Semana #	Resultado esperado
Sprint 1					
1	Investegicación preliminar sobre contenido de DevOps	40	5	1	Biblioteca de referencias de información de conocimiento DevOps creada para fines de selección
2	Selección de fuentes primarias y secundarias de información	16	2	2	Fuentes primarias y secundarias seleccionadas como fuente de verdad y referencia del contenido a documentar
3	Organización y comparación de fuentes seleccionadas	16	2	2	Fuentes de información categorizadas y delimitadas
4	Diagramación preliminar de estructura de contenido	40	5	3	Estructura de contenido diseñada de forma preliminar
Sprint 2					
5	Creación de estructura esqueleto de contenido para base de conocimiento	8	1	4	Esqueleto de contenido (temas, sub-temas, referencias) creado para base de conocimiento
6	Creación de estructura esqueleto de contenido para guía de adaptación	8	1	4	Esqueleto de contenido (temas, sub-temas, referencias) creado para guía de adaptación
7	Revisión y refinamiento preliminar de esqueletos de contenido	8	1	4	Limpieza y simplificación de los esqueletos de contenido
Sprint 3					
8	Descarga de contenido a esqueleto de base de conocimiento	56	7	5	Contenido plasmado en la base de conocimiento y su esqueleto
9	Refinamiento de contenido de base de conocimiento	16	2	6	Segunda iteración de descarga de contenido, para fines de simplificación y refinamiento
10	Revisión preliminar de contenido de base de conocimiento	8	1	6	Revisión de sintaxis, gramatical y de referencias, del contenido documentado
Sprint 4					
11	Descarga de contenido a esqueleto de guía de adaptacion	56	7	7	Contenido plasmado en la guía de adaptacion y su esqueleto
12	Refinamiento de contenido de guía de adaptación	16	2	8	Segunda iteración de descarga de contenido, para fines de simplificacion y refinamiento
13	Revisión preliminar de contenido de guía de adaptación	8	1	8	Revisión de sintaxis, gramatical y de referencias, del contenido documentado
Sprint 5					
14	Revisión gramatical y de sintaxis de contenido de base de conocimiento	8	1	9	Contenido revisado gramaticalmente
15	Revisión gramatical y de sintaxis de contenido de guía de adaptación	8	1	9	Contenido revisado gramaticalmente
16	Revisión final de contenido de base de conocimiento	8	1	9	Ajustes finales de contenido
17	Revisión final de contenido de guía de adaptación	8	1	9	Ajustes finales de contenido
Sprint 6					
18	Diseño gráfico y editorial de guía de adaptacion	32	4	10	Guía de adaptación diseñada en formato editorial
19	Diseño página web para base de conocimiento	32	4	10	Página web de base de conocimiento disponible
20	Revisión final general y ajustes de calidad	8	1	10	Ajustes y revisiones finales
	Total	400	50	10	

Tabla 5 Cronograma de actividades, Diseño de guía de adaptación de prácticas DevOps

Fuente: Proyecto Final - Cronograma de actividades

9.4 Factibilidad económica

El costo asociado a la creación de contenido se deriva principalmente del factor horas de recursos humanos, uso de una plataforma como servicio para la disposición de la página web asociada a la base de conocimiento y del proveedor de edición editorial para la guía de adaptación.

9.4.1 Presupuesto de recursos humanos

Rol	Cantidad	Cantidad Horas	Salario Hora (USD)	Costo Total (USD$)
Ingeniero DevOps	1	400	25$	10,000$
Diseñador Gráfico	1	100	15$	1,500$
Consultor Creador de Contenido	1	40	10$	400$
Total	**3**	**540**	**50$**	**11,900$**

Tabla 6 | Presupuesto, Recursos Humanos. Costos por hora

9.4.2 Presupuesto de plataformas y recursos tecnológicos

Recurso	Costo por Usuario (USD)	Cantidad Usuarios	Costo Mensual (USD)	Costo Anual (USD$)
GitHub Teams Plan	4$	3	12$	144$
Netlify Pro Plan	19$	3	57$	684$
Issuu Starter Plan	19$	3	57$	684$
Total	**42$**	**3**	**50$**	**1,512$**

Tabla 7 | Presupuesto Plataformas y Recursos tecnológicos. Costos por usuario (Diario, Mensual, Anual)

* Los costos de plataformas y recursos pueden ser considerados en los planes gratuitos si las limitaciones impuestas por cada proveedor no representan un impacto negativo en la productividad del equipo y la disponibilidad de los recursos asociados.

9.4.3 Retorno de Inversión (ROI)

La naturaleza de este proyecto final radica principalmente en el principio de exposición y compartimiento de información relacionada al movimiento cultural

denominado como DevOps. Mientras que el retorno de inversión enfocado en la entidad productora del contenido en cuestión es relativamente nulo, su propósito desde su concepción es simplemente crear conciencia sobre estos nuevos términos y conceptos mientras se expanden estas prácticas de manera natural en la industria de tecnologías de la información. De manera resumida, el real retorno de inversión radica en el éxito del objetivo general documentado en este proyecto de grado y no necesariamente radica en términos económicos factibles.

10. Casos de éxito

10.1 Soluciones GBH

GBH[12] es una compañía de soluciones tecnológicas en República Dominicana, con mayor incidencia en los servicios de desarrollo de software customizado y operaciones de TI. Fundada en el 2005, cuenta con más de 18 años de experiencia en una gran variedad de industrias y áreas de interés. Actualmente posee más de 150 colaboradores y se posiciona como una de las organizaciones de más prestigio a nivel nacional.

Su viaje evolutivo de DevOps empezó en el 2018 al convertirse en la primera organización en República Dominicana en prestar suficiente atención como para introducir entre sus rangos el rol de un ingeniero DevOps; justamente para liderar la transformación cultural, de automatización, y procesos que necesitaba.

Debido a la naturaleza y los objetivos del negocio particulares de GBH, la normalización y estandarización de tecnologías no fue un objetivo fácil de lograr dado que los servicios que ofrece se adaptan a las realidades y necesidades de otros clientes. Sin embargo, se identificaron cuáles eran los "pain points" comunes para tanto el proveedor (GBH) como el cliente.

Uno de los primeros esfuerzos que tuvieron sentido fue la creación de un conjunto de scripts a los cuales se les llamó "Dockerized" por tener una dependencia con Docker, el motor de contenerización más popular actualmente. Casi en paralelo, se trabajó con la primera implementación de pipelines de integración y despliegue. El éxito de estas

[12] gbh.com.do

iniciativas rápidamente captó la atención de todos los equipos y de los líderes de ingeniería. Aquí fue donde empezó la transformación cultural de GBH.

10.1.1 Dockerized (DEMS)

Dockerized, y su posterior evolución DEMS (que viene de: Development Environment Management System, Sistema de gestión de ambientes de desarrollo en español), fue una de las primeras demostraciones de éxito de manera aislada que tuvo GBH para uno de sus más importantes equipos en el 2018. Este éxito esparció rápidamente a otras personas y clientes, lo que facilitó la inversión en iniciativas de estandarización y automatización más adelante.

¿Qué exactamente logró Dockerized en sus inicios? Pues, de manera resumida, logró reducir el tiempo de onboarding al proyecto de 4-5 días en promedio, a solo 30 minutos. Estamos hablando de una reducción del 8000% del tiempo que originalmente se requería (Adames, 2022).

La situación particular que logró intervenir Dockerized, partió de los siguientes "pain points" para el equipo y la organización en general:

- El onboarding técnico al proyecto tomaba una semana de trabajo compartido. Los miembros existentes tenían que instruir al nuevo miembro sobre los requerimientos, dependencias y configuraciones que necesitaría instalar antes de correr la aplicación, y asistir en cualquier dificultad que éste encontrase. Todo era manual y no existía ningún tipo de documentación.
- La mayor parte del tiempo de onboarding se desperdiciaba en resolución de inconvenientes en la configuración de ambiente de desarrollo. Es decir, el desarrollador invertía días de trabajo tan solo logrando hacer funcionar su ambiente local de trabajo en su computador.
- La aplicación solo era configurable a través de un snapshot de una máquina virtual. Dicho snapshot carecía de instrucciones o documentación de uso, por lo que una persona sin previa experiencia en VMs se le haría cuesta arriba garantizar la funcionalidad sin asistencia de algún miembro equipo.

Tomando en cuenta todos estos puntos, se decidió aprovechar los conceptos básicos de scripting y la tecnología de contenerización para lograr potabilizar el sistema y que su configuración fuera un asunto tan trivial como la ejecución de un comando. Y así fue: Con tan solo ejecutar un script y un poco tiempo de espera, ya era posible aprovisionar un ambiente de desarrollo totalmente funcional de manera automatizada.

10.1.2 Integración continua (CI) para aplicaciones móviles

Tras el indiscutible éxito de Dockerized (DEMS), se logró afianzar aún más la confianza en las iniciativas orientadas en prácticas DevOps. Así fue como surgieron las primeras implementaciones de pipelines de integración continua y despliegues automatizados. Un hito importante alcanzado gracias a CI fue la configuración del primer pipeline para aplicaciones móviles.

Aún en el 2018, la información disponible para los ambientes de desarrollo móvil era bastante escasa. Sin embargo, GBH tenía una deficiencia en sus procesos de desarrollo y aseguramiento calidad para aplicaciones móviles.

En equipos compuestos por más de un desarrollador, y con capacidad para un solo ingeniero de QA, la velocidad de desarrollo no compensaba la velocidad de prueba, sobre todo cuando las compilaciones y simulaciones eran totalmente manuales. Por día (8 horas), un ingeniero QA solo podía probar máximo 4 cambios; mientras que los ingenieros de desarrollo producían casi el triple de esta cantidad.

El problema era evidente: compilaciones manuales. GBH entonces diseñó un flujo automatizado en el que se logró que la compilación se realizara en demanda por nodos aprovisionados en AWS, y los artefactos requeridos fueran almacenados y expuestos a través de códigos QR, lo que facilitaría el proceso de instalación para los ingenieros QA.

Con esto, se logró una optimización de aproximadamente el 300% en velocidad de prueba. También se obtuvo un beneficio adicional: el equipo completo podía instalar el mismo artefacto y validar cualquier defecto o error encontrado.

10.2 Netflix

A pesar de que Netflix es una empresa de entretenimiento, no se ha quedado rezagada con respecto a varias de las principales empresas de TI en términos de avances tecnológicos. Netflix ha tenido un gran impacto en el mundo tecnológico a lo largo de los años con su única aplicación de transmisión de video, esfuerzos de ingeniería de clase mundial, cultura y desarrollo de productos.

Según un artículo que muestra un poco sobre la evolución de DevOps en Netflix, DevOps es uno de esos métodos que Netflix ejemplifica admirablemente. Su cultura DevOps les ha permitido desarrollarse más rápidamente, lo que se ha traducido en numerosos beneficios comerciales. También los ayudó a lograr un tiempo de disponibilidad casi perfecto, brindar nuevos servicios a los consumidores más rápidamente y aumentar sus suscriptores y horas de transmisión (Dhaduk, 2022).

Netflix es el servicio de transmisión más popular del mundo en la actualidad, con casi 214 millones de clientes y transmisión en más de 190 países. Gran parte de su éxito se puede atribuir a su capacidad para adaptar la tecnología más nueva y su cultura DevOps, que le permite desarrollarse rápidamente para satisfacer las necesidades de los consumidores y mejorar las experiencias de los usuarios. Sin embargo, Netflix no cree en DevOps.

10.2.1 Migración a la nube

Todo comenzó con la peor interrupción en la historia de Netflix cuando se enfrentó a una importante corrupción de la base de datos en 2008 y no pudo enviar DVD a sus miembros durante tres días. En ese momento, Netflix tenía aproximadamente 8.4 millones de clientes y un tercio de ellos se vieron afectados por la interrupción. Hizo que Netflix se mudara a la nube y le diera a su infraestructura un cambio de imagen completo. Netflix eligió a AWS como su socio en la nube y tardó casi siete años en completar su migración a la nube.

Netflix optó por reescribir toda la aplicación en la nube para volverse verdaderamente nativa de la nube, lo que cambió fundamentalmente la forma en que operaba la empresa. Como parte importante de su transformación, Netflix convirtió su aplicación Java

monolítica basada en el centro de datos en una arquitectura de microservicios Java basada en la nube.

Provocó los siguientes cambios:

- Modelo de datos desnormalizados usando bases de datos NoSQL[13].
- Permitió que los equipos de Netflix se acoplaran libremente.
- Permitió a los equipos crear e impulsar cambios a la velocidad con la que se sentían cómodos.
- Coordinación de liberación centralizada.
- Los ciclos de aprovisionamiento de hardware de varias semanas llevaron a una entrega continua.
- Los equipos de ingeniería tomaron decisiones independientes utilizando herramientas de autoservicio.

Como resultado, ayudó a Netflix a acelerar la innovación y tropezar con la cultura DevOps. Netflix también ganó ocho veces más suscriptores que en 2008. Y las horas mensuales de transmisión de Netflix también crecieron mil veces desde diciembre de 2007 hasta diciembre de 2015 (Izrailevsky, Vlaovic, & Meshenberg, 2016).

Después de completar su migración a la nube a AWS en 2016, Netflix tenía:

- Más de 100 de microservicios
- Más de 1,000 de cambios diarios en producción
- Más de 10,000 máquinas virtuales corriendo en AWS
- Más de 100,000 de interacciones de clientes por segundo
- Más de 1 millón de clientes
- Más de 2,500 millones de métricas entregadas, procesadas y almacenadas cada minuto
- Más de 10 billones de horas de transmisión cada trimestre

[13] NoSQL Explained

10.2.2 Lecciones aprendidas de Netflix

- **Centrarse en dar libertad y responsabilidad a los ingenieros**: Netflix tiene como objetivo contratar personas inteligentes y brindarles la libertad de resolver los problemas de la manera que mejor les parezca. Por lo tanto, no tiene que crear restricciones y barandillas artificiales para predecir lo que deben hacer sus desarrolladores. Pero en su lugar, contrate a personas que puedan desarrollar un equilibrio de libertad y responsabilidad.
- **No crear silos, paredes y cercas**: Los equipos de Netflix saben dónde encajan en el ecosistema, su trabajo con otros equipos, dependientes y dependencias.
- **Siempre ponga la satisfacción del cliente primero**: El objetivo final de DevOps es orientarse al cliente y centrarse en mejorar la experiencia del usuario con cada lanzamiento.
- **No haga DevOps, pero concéntrese en la cultura**: En Netflix, DevOps surgió como el maravilloso resultado de su cultura, pensamiento y prácticas saludables.

10.2.3 Datos curiosos

- El 21 de abril de 2011, AWS experimentó una gran interrupción en la región Este de EE. UU., pero la transmisión de Netflix funcionó sin interrupciones. Y el 24 de diciembre de 2012, AWS enfrentó problemas en los servicios del servicio de balanceo de carga, pero Netflix no experimentó un apagón inmediato. El sitio web de Netflix estuvo activo durante la interrupción y admitió la mayoría de sus servicios y transmisión, aunque con una latencia más alta en algunos dispositivos (Izrailevsky, Vlaovic, & Meshenberg, 2016).

11. Conclusiones

11.1 Conclusión general

A pesar de los retos inherentes a la adaptación de prácticas DevOps, como la resistencia al cambio y aquellos heredados de cualquier tipo transformación digital, el enfoque que promueven genera beneficios y cambios culturales significativos apoyándose en principios de automatización, medición e intercambio de información. En la última década, DevOps ha tenido un impacto y crecimiento exponencial por parte de organizaciones con un alto nivel evolutivo y se espera que éstos índices sigan creciendo aún más en el futuro.

Con los entregables de este proyecto final, se propone diseñar guías de conocimiento para cambiar los paradigmas tradicionales de entrega de valor y aplicar la mejora continua en todos los aspectos de los colaboradores y la organización objetivo en general.

La guía de adoptación reconoce que toda organización, tomando en cuenta su cultura e individuos, es totalmente única. Y, por tanto, sus problemas, necesidades y objetivos también lo son. Al alinear nuestra visión sobre cómo empieza este proceso de transformación, tomando en cuenta la información recolectada por diferentes fuentes patrocinados por el proyecto de investigación DORA en cuánto los factores de éxito de muchas organizaciones en el mundo, se logró crear una guía de adaptación de alto nivel de las prácticas de DevOps que es utilizable por organizaciones con procesos de ingeniería de software sin depender de su tamaño, contexto, industria y objetivos comerciales.

11.2 Resultados

Concluido el trabajo inicial presentado en este proyecto final, podemos afirmar que los objetivos específicos planteados fueron alcanzados exitosamente:

- Expusimos y exploramos diferentes casos de éxito, tanto de una compañía de alto desempeño internacional, como una con un menor alcanza a nivel de República Dominicana; en esta exposición se demuestra que las prácticas DevOps se aplican independientemente del tamaño de la organización y

generan beneficios directos e indirectos que promueven el crecimiento cultural y del negocio a gran escala.

- Se creó una base de conocimiento en formato digital publicada en plataformas que promueven la colaboración de código libre que está orientada a servir principalmente a profesionales relacionados a tecnologías de información.
- Se diseñó una guía de adaptación de prácticas de DevOps a alto nivel con una audiencia objetivo a nivel de empresas con procesos de ingeniería de software. En ésta se documentó, en base a la experiencia de los autores de este proyecto final y de los informes recolectados por el proyecto de investigación DORA de Google a través de la última década.

11.3 Recomendaciones

Hay dos ingredientes clave en una transformación efectiva y continua:

- procesos para ejecutar el cambio organizacional mediante el establecimiento de objetivos y permitiendo la experimentación en equipo,
- y mecanismos para difundir las buenas prácticas en toda la organización.

Por tanto, nuestro proyecto final ofrece dos recomendaciones finales:

11.3.1 Establecer objetivos claros

Antes de realizar cualquier implementación o cambio organizacional, se debe definir primero objetivos claros a perseguir. De esta manera, es posible segmentar un gran objetivo a entregables más pequeños. A través de la definición de estos objetivos, es posible seguir un modelo iterativo en el que primero los equipos entienden la dirección del cambio, entienden y evalúan su situación actual, definen una nueva situación objetivo e iteran ciclos de trabajo hasta alcanzar los objetivos planteados.

11.3.2 Permitir la experimentación

La experimentación es esencial para adoptar correctamente la cultura de DevOps en una organización. A través de esta, es posible generar resultados importantes que permiten dar visibilidad y confianza de los beneficios obtenidos a los pilares de toma de decisiones organizacionales (gerentes, directores, etc.).

12. Referencias

Adames, A. (4 de Mayo de 2022). *DevOps at GBH - How did we get started?* Obtenido de GBH: https://gbh.com.do/devops-at-gbh-how-did-we-get-started

agilemanifesto.org. (s.f.). *Manifesto for Agile Software Development.* Recuperado el 2 de Agosto de 2022, de agilemanifesto.org: https://agilemanifesto.org/

Allied Market Research. (s.f.). *DevOps Market by Component, Cloud Type, Organization Size, and Industry Vertical.* Recuperado el 9 de Agosto de 2022, de https://www.alliedmarketresearch.com/devops-market

Altvater, A. (17 de Septiembre de 2017). *What is Agile Methodology? How It Works, Best Practices, Tools.* Obtenido de Stackify: https://stackify.com/agile-methodology/

AWS. (s.f.). *Cloud computing with AWS.* Recuperado el 24 de Junio de 2022, de Amazon Web Services (AWS): https://aws.amazon.com/what-is-aws/

DevOps Agile Skills Association (DASA). (17 de Agosto de 2022). *A Beginner's Guide to DevOps.* Obtenido de https://www.devopsagileskills.org/a-beginners-guide-to-devops/

DevOps Research and Assessment LLC. (s.f.). *DORA's Research Program.* Recuperado el 10 de Julio de 2022, de https://www.devops-research.com/research.html

Dhaduk, H. (24 de Febrero de 2022). *How Netflix Became A Master of DevOps? An Exclusive Case Study.* Recuperado el 12 de Julio de 2022, de Sinform: https://www.simform.com/blog/netflix-devops-case-study/

DigitalOcean. (5 de Diciembre de 2017). *An Introduction to Metrics, Monitoring, and Alerting.* Recuperado el 10 de Junio de 2022, de https://www.digitalocean.com/community/tutorials/an-introduction-to-metrics-monitoring-and-alerting

Edwards, D. (18 de Septiembre de 2012). *The Short History of DevOps.* Obtenido de https://www.youtube.com/watch?v=o7-IuYS0iSE

ExploringBits. (4 de Enero de 2021). *Importance of software engineering.* Obtenido de ExploringBits: https://exploringbits.com/importance-of-software-engineering/

GitLab. (11 de Febrero de 2022). *4 Must-know DevOps principles.* Obtenido de https://about.gitlab.com/blog/2022/02/11/4-must-know-devops-principles/

GlobalNewswire. (2021). *DevOps Market Size is Expected to Reach USD 37,227 Million by 2030 with CAGR 20% Report by Acumen Research and Consulting.* Obtenido de https://www.globenewswire.com/en/news-release/2022/06/23/2467580/0/en/DevOps-Market-Size-is-Expected-to-Reach-USD-37-227-Million-by-2030-with-CAGR-20-Report-by-Acumen-Research-and-Consulting.html

Google. (s.f.). *Accelerate State of DevOps 2021.* Recuperado el Mayo de 2022, de https://services.google.com/fh/files/misc/state-of-devops-2021.pdf

Google. (s.f.). *Google Cloud.* Recuperado el Julio de 2022, de Google Cloud Documentation: https://cloud.google.com/docs/overview

Google. (s.f.). *State of DevOps 2019.* Recuperado el 5 de Abril de 2022, de https://services.google.com/fh/files/misc/state-of-devops-2019.pdf

Haight, C. (27 de October de 2010). *DevOps: Born in the Cloud and Coming to the Enterprise.* Obtenido de https://www.gartner.com/en/documents/1458129

Izrailevsky, Y., Vlaovic, S., & Meshenberg, R. (11 de Febrero de 2016). *Completing the Netflix Cloud Migration.* Obtenido de https://about.netflix.com/en/news/completing-the-netflix-cloud-migration

León, G. (19 de Mayo de 2018). *Psicología del color aplicada al Marketing.* Obtenido de MailClick: https://www.mailclick.com.mx/psicologia-del-color-aplicada-al-marketing-digital

Loukides, M. (7 de Junio de 2012). *What is DevOps?* Obtenido de O'Reilly Radar: http://radar.oreilly.com/2012/06/what-is-devops.html

Markdown Guide. (s.f.). *Getting Started.* Recuperado el 17 de Julio de 2022, de https://www.markdownguide.org/getting-started/

Martin, M. (18 de Junio de 2022). *What is Software Engineering? Definition, Basics, Characteristics.* Obtenido de Guru99: https://www.guru99.com/what-is-software-engineering.html

Mezak, S. (25 de Enero de 2018). *The Origins of DevOps: What's in a Name?* Obtenido de DevOps.com: https://devops.com/the-origins-of-devops-whats-in-a-name/

Microsoft Azure. (s.f.). *What is Azure?* Recuperado el 14 de Marzo de 2022, de Azure: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/

Puppet & CircleCI. (s.f.). *State of DevOps 2020*. Recuperado el 28 de Abril de 2022, de https://media.webteam.puppet.com/uploads/2020/11/Puppet-State-of-DevOps-Report-2020.pdf

Puppet. (s.f.). *State of DevOps 2019*. Recuperado el 6 de Abril de 2022, de https://media.webteam.puppet.com/uploads/2019/11/Puppet-State-of-DevOps-Report-2018_update.pdf

Puppet. (s.f.). *State of DevOps 2021*. Recuperado el 6 de Abril de 2022, de https://media.webteam.puppet.com/uploads/2021/07/Puppet-State-of-DevOps-Report-2021.pdf

Raza, M. (11 de Agosto de 2021). *IT Operations (ITOps) Roles & Responsibilities*. Obtenido de BMC Blogs: https://www.bmc.com/blogs/it-operations-roles/

RedHad. (s.f.). *What are cloud services?* Recuperado el 24 de Abril de 2022, de https://www.redhat.com/en/topics/cloud-computing/what-are-cloud-services

Smith, M. (31 de Agosto de 2021). *Why is software engineering important: 7 key reasons it's vital*. Obtenido de Developer Pitstop: https://developerpitstop.com/why-is-software-engineering-important/

Splunk. (s.f.). *What Is Observability?* Recuperado el 12 de Agosto de 2022, de https://www.splunk.com/en_us/data-insider/what-is-observability.html

Studio, O. (19 de Noviembre de 2019). *Psicología del color en el diseño web*. Obtenido de onion.st: https://www.onion.st/psicologia-del-color-en-el-diseno-web

Tricentis. (28 de Julio de 2022). *What is Continuous Testing?* Obtenido de https://www.tricentis.com/products/what-is-continuous-testing/

13. Anexos

13.1 Enlaces de la base de conocimiento a la fecha 2022-09-01

- **Página de Inicio (Ilustración 10)**: contiene una vista general de los módulos más importantes y un botón enlazado a la documentación principal, visible en: https://devops-kb.netlify.app.
- **Sección 'Antes de Empezar' (Ilustración 11):** contiene una breve introducción a la base de conocimiento en conjunto con recomendaciones generales de uso y navegación, motivación y programa de contenido, visible en: https://devops-kb.netlify.app/docs/antes-de-empezar/bienvenida.
- **Sección 'Introducción' (Ilustración 12):** introduce formalmente DevOps y la base de conocimiento y expande brevemente en su alcance y estructura, visible en: https://devops-kb.netlify.app/docs/introduccion/intro-base.
- **Sección 'Fundamentos de DevOps' (Ilustración 13):** esta sección abunda sobre el término de DevOps, su cultura, herramientas, procesos, prácticas y principios, incluyendo su origen, visible en: https://devops-kb.netlify.app/docs/fundamentos-de-devops/introduccion-a-devops.
- **Sección 'Computación en la nube' (Ilustración 14):** incluye el contexto general de los servicios y proveedores en la nube, visible en: https://devops-kb.netlify.app/docs/computacion-en-la-nube/servicios-en-la-nube.
- **Sección 'Ciclo de Desarrollo' (Ilustración 15):** elabora en lo que compete al ciclo de vida de desarrollo de software (SLDC), visible en: https://devops-kb.netlify.app/docs/ciclo-de-desarrollo/sdlc.
- **Sección 'Contenerización' (Ilustración 16):** define los conceptos técnicos asociados a la contenerización y el empaquetamiento de aplicaciones, visible en: https://devops-kb.netlify.app/docs/contenerizacion/contenedores.
- **Sección 'Integración continua' (Ilustración 17):** contiene información sobre la integración continua, sus herramientas y mejores prácticas, visible en: https://devops-kb.netlify.app/docs/integracion-continua/integracion-continua.
- **Sección 'Entrega continua' (Ilustración 18):** introduce formalmente DevOps y la base de conocimiento y expande brevemente en su alcance y estructura, visible en: https://devops-kb.netlify.app/docs/entrega-continua/entrega-continua.

- **Sección 'Infraestructura' (Ilustración 19):** contiene prácticas enfocadas a infraestructura propias de la cultura DevOps, visible en: https://devops-kb.netlify.app/docs/infraestructura-como-codigo/aprovisionamiento-de-infraestructura.
- **Sección 'Monitoreo y alertas' (Ilustración 20):** expande en los conocimientos relacionados a la observabilidad, el monitoreo de infraestructura y aplicaciones y alertas, visible en: https://devops-kb.netlify.app/docs/monitoreo-y-alertas/intro.
- **Sección 'Aseguramiento de la calidad' (Ilustración 21):** contiene información sobre el concepto de pruebas continuas y automatizadas, visible en: https://devops-kb.netlify.app/docs/aseguramiento-de-calidad/intro.
- **Sección 'Seguridad continua' (Ilustración 22):** incluye en las prácticas DevOps el concepto de seguridad de la información, visible en: https://devops-kb.netlify.app/docs/seguridad-continua/devsecops.

13.2 Guía de adaptación de prácticas DevOps

(Ver Anexos adjuntos.)

Resumen ejecutivo

En la última década, mucho se ha hablado y expandido sobre lo qué es DevOps y lo que realmente significa. Se han entrevistado y consultado miles de profesionales alrededor de los años tratando de encontrar patrones de éxito comunes que puedan aplicarse o referenciarse de manera genérica y a alto nivel. Por igual, se ha explorado las relaciones existentes entre el rendimiento de TI, las prácticas de DevOps, la cultura, el rendimiento organizacional y otros elementos que afectan los resultados del negocio.

Con toda esta información, se ha producido una referencia común al éxito de organizaciones que han implementado de manera exitosa las prácticas DevOps y han evidenciado de primera mano las oportunidades que trae sobre la mesa.

Hemos aprendido. Y este aprendizaje se demuestra en nuestra guía de de adaptación de metodologías DevOps, donde se describe el proceso evolutivo a través de varias etapas y prácticas críticas que ayudarán a una organización a alcanzar el éxito con estas nuevas prácticas revolucionarias.

Audiencia objetivo

La audiencia objetivo de esta guía de implementación no es más que un prospecto ideal que promete corresponder de manera más armoniosa las diferentes etapas y recomendaciones que se detallarán en lo adelante. Sin embargo, la audiencia objetivo solo represeta una figura ideal; en la realidad cualquier organización o entidad, con procesos de ingeniería de software internos y con dependencia a sistemas o aplicaciones en desarrollo continuo, es capaz de sacar provecho a toda la información que en este documento se expone.

Recomendamos que, independientemente de la realidad de tu organización, consultes esta guía, puesto puede arrojar resultados interesantes que aportar a la mejora continua de los procesos y cultura organizacional.

DevOps Introducción

Antes de entrar en materia sobre cómo adoptar las metodologías asociadas a DevOps para tu organización, es importante empezar desde el principio y responder preguntas fundamentales para afianzar los objetivos que se desean lograr.

Definición

Bien, entonces, ¿Qué Wes DevOps?

De manera simple:

DevOps no es más que Desarrollo y Operaciones trabajando en conjunto como una única unidad; donde tratan continuamente de reducir complejidad, eliminar esfuerzos innecesarios y trabajo manual, repetitivo y tedioso, compartir principios culturales y técnicos, todo esto utilizando las metodologías ágiles como su vehículo principal.

En un contexto más abierto:

DevOps es un movimiento cultural que intenta romper sobre los Wya establecidos paradigmas de Desarrollo y Operaciones en cuanto a objetivos de negocio, indicadores de rendimiento, gestión del trabajo, y entrega de valor y resultados.

DevOps como movimiento o metodología se basa en varios principios culturales que discutiremos en detalle más adelante. Pero antes, entendamos de dónde surge DevOps.

Origen

Antes de definir de manera puntual lo que es DevOps, es importante describir el origen de DevOps debemos hablar primero de Patrick Debois, a quién se le atribuye más comúnmente ser el padre del término DevOps.

Nuestra historia empieza en 2008 donde Patrick presentó por primera vez el concepto de "infraestructura ágil" en una conferencia en Toronto llamada Agile 2008. En 2009, organizó la primera conferencia relacionada a DevOps llamada **DevOpsDays** donde el objetivo era traer a personas de desarrollo y operaciones a trabajar en conjunto.

Inicialmente el tema de la conferencia iba a ser "Administración de Sistemas Ágil" (Agile System Administration) pero para Patrick esto era in título muy largo, así que como el enfoque de la conferencia sería orientada a Desarrollo y Operaciones trabajando en conjunto, combinó estas dos palabras obteniendo entonces DevOps. Nunca hubo una gran intención o propósito detrás de dicho nombre, solo se quería un nombre más corto.

De acuerdo con el vídeo **"The Short History of DevOps"**, luego de la primera instancia de DevOpsDays, las personas que participaron siguieron conversando sobre este tema en diferentes lugares de la Internet, principalmente en Twitter. Una de las razones del rápido esparcimiento de la palabra DevOps se debió a que en Twitter las discusiones relacionadas a la conferencia fueron etiquetadas con el hashtag *#DevOpsDays*, pero debido a las limitantes de caracteres de Twitter por cada tuit en ese entonces, el hashtag evolucionó a ser simplemente *#DevOps*.

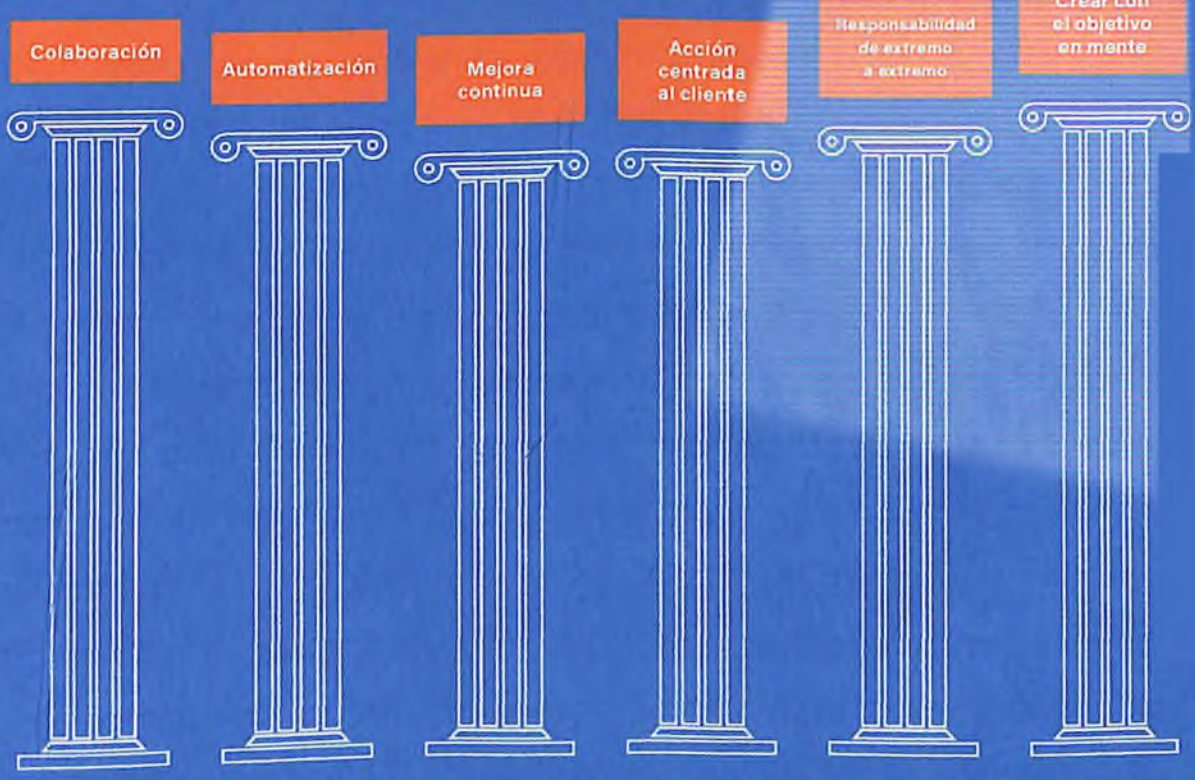
Principios culturales

Es importante mencionar que no existe una forma única de implementar DevOps. DevOps posee un valor único e íntimo para la organización que le adopte.

Sin embargo, podemos asociar DevOps a varios principios o pilares que permiten, sin importar el contexto, entorno, responsables, u organización, identificar si se cumple o no con la metodología DevOps.

Los principios de DevOps no son una especificación o estándar. Es bueno verlos como un marco de referencia más que leyes o normativas.

A modo resumen, los principios de DevOps son:



A continuación describiremos más a detalle de que se trata cada uno.

Colaboración

La premisa clave detrás de DevOps es la colaboración entre todos los equipos que forman parte del ciclo de desarrollo, incluyendo, pero no limitándose a desarrollo y operaciones. Esto quiere decir, entre otras cosas, que se trabaja como un solo equipo durante todo el ciclo de vida de desarrollo.

Debido a la responsabilidad compartida sobre la entrega de valor al producto, se requiere de un equipo multidisciplinario en donde existan altos niveles de propiedad y frecuente comunicación.

Mejora continua

La mejora continua se estableció como un elemento básico de las prácticas ágiles. Es la práctica de centrarse en la experimentación, minimizar el desperdicio y optimizar la velocidad, el costo y la facilidad de entrega.

La mejora continua también está ligada a la entrega continua, lo que permite a los equipos de DevOps impulsar continuamente actualizaciones que mejoran la eficiencia de los sistemas de software.

Acción centrada al cliente

Los equipos de DevOps utilizan cortos ciclos de retroalimentación con clientes y usuarios finales para desarrollar productos y servicios centrados en las necesidades de estos.

Las prácticas de DevOps permiten una recopilación y respuesta rápidas a los comentarios de los usuarios mediante el uso de monitoreo en tiempo real y despliegues rápidos y frecuentes.

Los equipos obtienen visibilidad inmediata de cómo los usuarios reales interactúan con las aplicaciones y utilizan esa información para trabajar en mejoras adicionales.

Responsabilidad de extremo a extremo

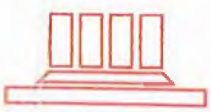
La responsabilidad de extremo a extremo significa que todo el equipo es el dueño del resultado y no hay discriminación ni señalización por culpa. Actuar como equipo y asumir el fracaso y el éxito como una entidad única permite a los colaboradores sentirse parte de una meta mayor que, a su vez, produce altos niveles de propiedad, motivación y resistencia.

No significa que no haya responsabilidades y límites claros, el beneficio real de este principio es garantizar que todo el equipo sea capaz de comprender el impacto de su trabajo en las responsabilidades de los demás; asegurando así que haya una sana rendición de cuentas y transparencia.

Crear con el objetivo en mente

Este principio implica comprender las necesidades de los clientes y crear productos o servicios que resuelvan problemas reales.

Los equipos no deben "construir en una burbuja" o crear software basado en suposiciones sobre cómo los consumidores usarán el software. Más bien, los equipos de DevOps deben tener una comprensión holística del producto, desde la creación hasta la implementación.



Procesos y prácticas comúnes

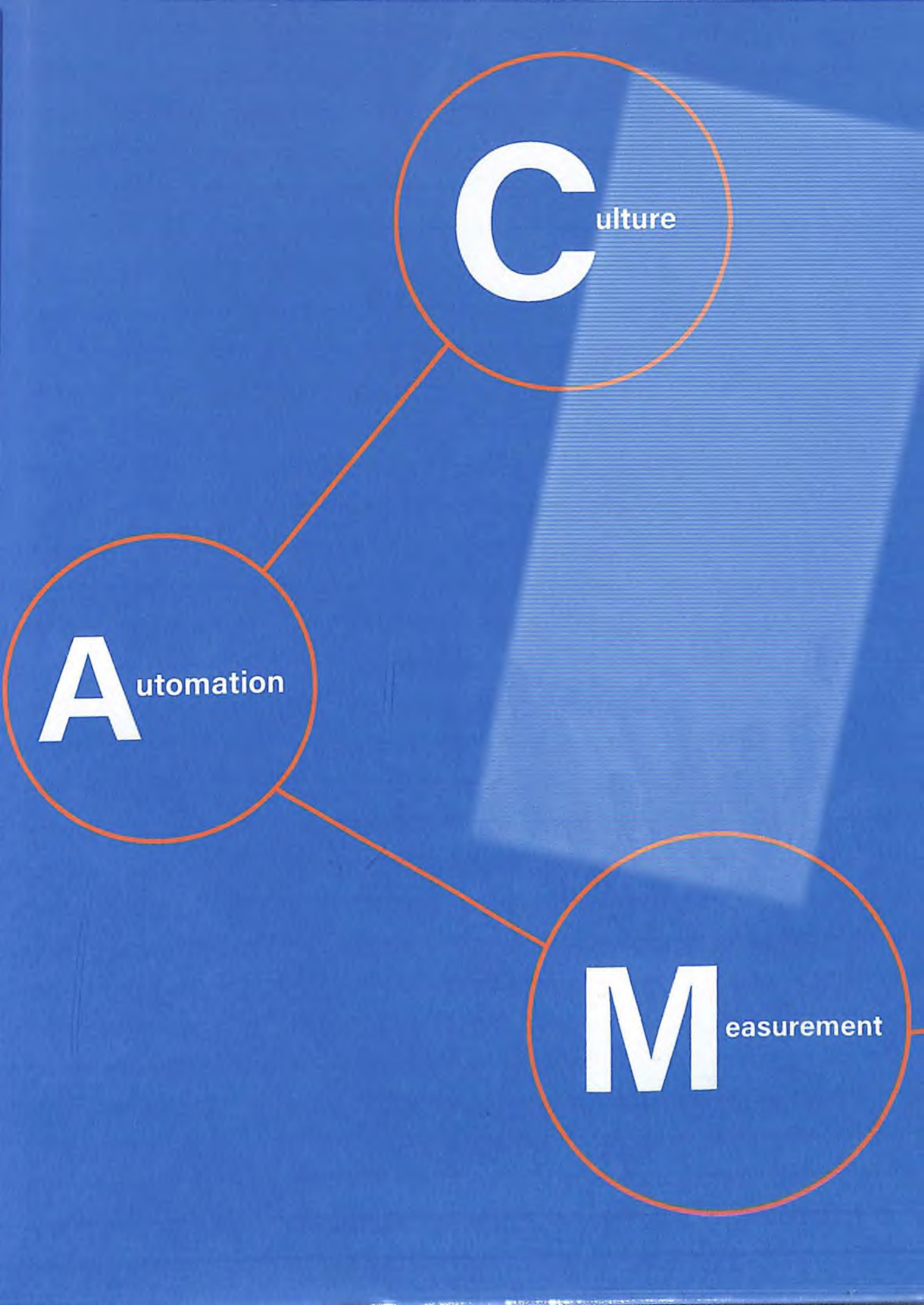
En la mayoría de los recursos existentes sobre DevOps es fácil notar que hay muchas variaciones del entendimiento de lo que son los procesos de DevOps. En factor común, podemos notar que siempre se destacan los siguientes procesos:

- **Integración continua**
- **Entrega continua**

Lo cierto es que no existe una fórmula oficial sobre lo que realmente compete a procesos de DevOps, sin embargo, tras consolidar diferentes fuentes y recursos, y a su vez extender su alcance, en este curso trataremos los siguientes procesos:

- **Flujo de desarrollo**
- **Aprovisionamiento de infraestructura automatizada**
- **Integración continua**
- **Entrega continua**
- **Despliegue continuo**
- **Pruebas continuas**
- **Monitoreo y alertas**
- **Seguridad integrada**

En la sección de catálogo de capacidades abundaremos sobre cada uno de estos procesos de manera detallada.


Culture
Automation
Measurement

Modelo CAMS

Una hipótesis popular en el proceso de adopción de metodologías DevOps en organizaciones es que la gran mayoría de las iniciativas exitosas se basan en focos de éxito existentes dentro de uno o más equipos y que, por el contrario, las iniciativas de arriba hacia abajo (top-down) sin éxito previo a nivel de equipo tienden a fallar.

La hipótesis se basa en la experiencia de los autores del informe **Puppet State of DevOps 2018**. En dicho informe, los autores afirman que han trabajado con múltiples organizaciones en las que han observado distintos patrones de exito.

Las prácticas de DevOps aún son relativamente nuevas en comparación con la edad de la mayoría de las grandes empresas, y es de esperarse que se requiera tiempo, esfuerzo y disciplina para generar cambios en una organización.

En el mismo informe, los autores afirman que han visto varias iniciativas de arriba hacia abajo relacionadas con TI, a menudo etiquetadas como 'DevOps', que no lograron brindar mejoras fundamentales, como la capacidad de brindar servicios de TI más rápido con niveles más altos de calidad.

Es por esto que es importante medir los resultados de las iniciativas relacionadas con TI en lugar de basarnos en anécdotas, especialmente en vista de las perspectivas muy diferentes de las que provienen las personas que relatan estas anécdotas. Por tanto, nos apoyaremos del modelo CAMS, un marco ampliamente aceptado para DevOps. Originalmente acuñado por *Damon Edwards* y *John Willis*, CAMS significa cultura, automatización, medición y compartir.

Los elementos de estos acrónimos son, por supuesto, categorías amplias sin límites formales. Pero la definición de CAMS ha demostrado ser un modelo útil y factible durante muchos años, especialmente porque DevOps ha evolucionado hacia nuevas áreas, como operaciones de red y seguridad.

Existe una correlación directa entre las organizaciones que han logrado un progreso concreto en estas cuatro áreas y haber madurado la adopción DevOps a niveles aceptables.

Evaluación comparativa

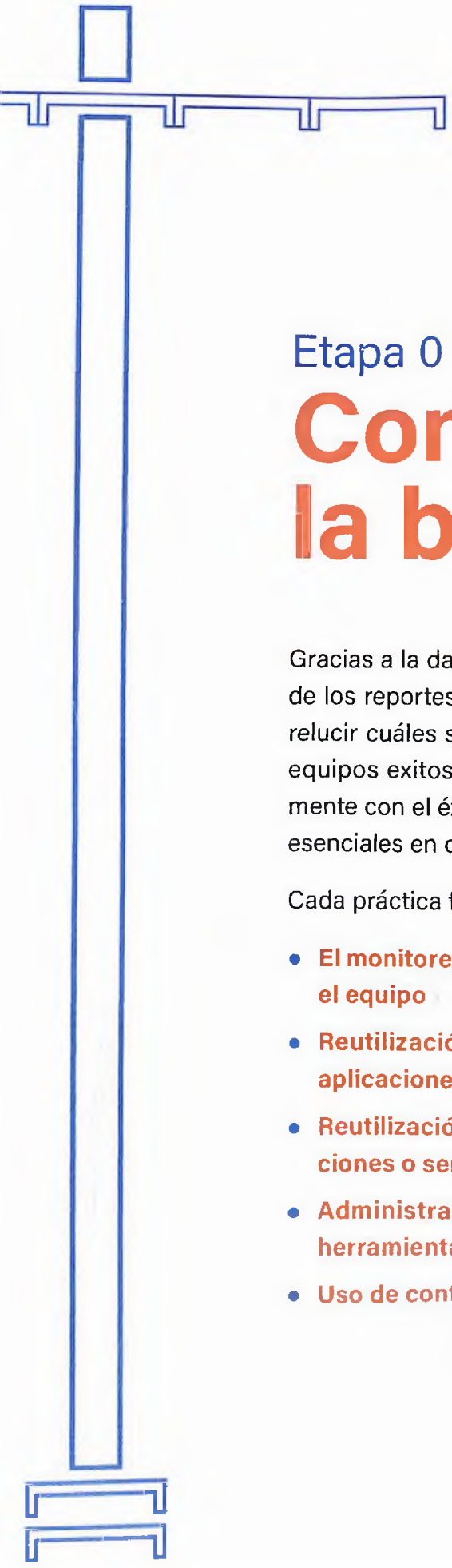
CAMS es un modelo útil para comparar el progreso evolutivo de su organización.

Los datos expuestos en varios reportes de estado de DevOps muestran que las organizaciones altamente evolucionadas:

Cada organización comienza la evolución en DevOps desde un lugar único. Tiene tecnologías desfasadas o antiguas, formas establecidas de hacer las cosas, su propia misión comercial específica y su propia cultura particular. Por lo tanto, no existe una ruta única para una transformación DevOps; en cambio, hay muchos caminos evolutivos posibles.

A partir de las investigaciones realizadas por estos mismos reportes, se revelaron cinco (5) etapas de la evolución de DevOps y también un conjunto de prácticas fundamentales que son críticas a lo largo de un viaje evolutivo de DevOps. En esta guía de adaptación no reinventaremos la rueda, pero extenderemos estas etapas iniciales a aprendizajes comunes de varias organizaciones y factores de éxito comunes.

Etapas de evolución de DevOps



Etapa 0

Construir la base

Gracias a la data recolectada por diferentes fuentes, en especial de los reportes del estado de DevOps de todos los años, sale a relucir cuáles son las prácticas fundamentales que emplean los equipos exitosos. Estas prácticas se correlacionan tan fuertemente con el éxito de DevOps que podemos determinar que son esenciales en cualquier etapa del desarrollo de DevOps.

Cada práctica fundamental se puede describir en una frase:

- **El monitoreo y las alertas es configurable y visible por todo el equipo**
- **Reutilización de los patrones de despliegue para crear aplicaciones o servicios**
- **Reutilización de los patrones de prueba para crear aplicaciones o servicios**
- **Administración de las configuraciones a través de una herramienta de gestión configuración**
- **Uso de control de versiones**

Cuando examinamos cada una de estas prácticas más de cerca, encontramos que las organizaciones altamente evolucionadas tenían muchas más probabilidades de usarlas que las organizaciones menos evolucionadas.

En conclusión, las prácticas fundamentales enumeradas anteriormente son parte integral de DevOps y críticas para el éxito de DevOps en cualquier etapa en la que se encuentre una organización.

Las prácticas fundamentales siguen siendo importantes incluso para aquellas organizaciones que ya han mostrado éxito en DevOps.

Todo se base en compartir

Al estudiar las prácticas fundamentales reveladas por diversas investigaciones de DevOps y sus casos de éxito en una gran variedad de organizaciones, nos damos cuenta que todas dependen y promueven el compartir. Todas las prácticas fundamentales permiten o dependen de compartir, lo que nos dice que la clave para escalar el éxito de DevOps es justamente la adopción de prácticas que promuevan el compartir.

Tiene sentido: cuando las personas ven algo que va bien, quieren replicar ese éxito y, por supuesto, quieren compartir sus éxitos. Digamos que su equipo ha desplegado con éxito una aplicación 10 veces, y también digamos que este tipo de despliegue normalmente ha causado muchos problemas a su equipo (y a otros). Lo más probable es que alguien se dé cuenta y quiera saber cómo lo estás haciendo. Así es como las prácticas de DevOps comienzan a expandirse en varios equipos.

La medición es una pieza central de DevOps

El monitoreo no es solo para operaciones o para algún equipo de DevOps recién creado: es para todos los equipos que trabajan con tecnología. La adopción del monitoreo para todos los equipos subraya uno de los puntos más importantes de DevOps: no necesita crear un nuevo equipo con nuevos superpoderes, sino que debe empoderar a todos los equipos existentes para que puedan trabajar juntos de nuevas maneras.

Independientemente de las circunstancias específicas, el monitoreo y las alertas como autoservicio son una contramedida para el viejo patrón anti-desarrollo y operaciones que trabajan en silos.

El simple hecho de abrir el acceso a estas métricas clave permite:

una cultura de intercambio y compartir,

llena los bucles de retroalimentación,

permite la retroalimentación continua y,

promueve una cultura de aprendizaje continuo entre los equipos.

Inculcar el sentido de propiedad y la responsabilidad mediante el empoderamiento de los equipos de entrega de servicios para recopilar, compartir y personalizar los datos de monitoreo es una parte fundamental del cambio cultural de DevOps y permite cambios aún más fundamentales más adelante en el proceso.

Etapa 1

Normalizar la pila de tecnología

La mayoría de las organizaciones que utilizan cualquier cantidad de tecnología están lidiando con mucha complejidad, lo que ralentiza sus esfuerzos para hacer avanzar el negocio. Por lo tanto, los primeros esfuerzos en una transformación DevOps (o cualquier tipo de transformación comercial) se centran en reducir la complejidad.

Prácticas que definen la Etapa 1

Para lograr reducir la complejidad y normalizar la pila de tecnología, nos basaremos en prácticas esenciales que definen la Etapa 1:

Uso de control de versiones por los equipos de desarrollo

Uso estándar de sistemas operativos para despliegues

Exploremos más a detalle el alcance de estas prácticas.

Uso de control de versiones por los equipos de desarrollo

El uso del control de versiones por parte de los equipos de desarrollo de aplicaciones representa un cambio fundamental en la forma en que los equipos producen software. Es el primer paso para implementar la integración continua en el camino hacia la entrega continua, que suele ser el objetivo de una iniciativa DevOps.

Cuando los equipos de aplicaciones usan el control de versiones, generalmente producen código desplegable con mucha más frecuencia que antes. Por lo tanto, aumenta la presión sobre los equipos de operaciones para que se realicen los despliegues rápidamente mientras mantienen los sistemas seguros y estables.

Uso estándar de sistemas operativos para despliegues

En las primeras etapas de una evolución de DevOps debe existir un esfuerzo en normalizar la pila tecnológica y eliminar los recursos que deben ser mantenidos, probados y manejados como casos únicos, aislados o antiguos.

Cuanta más variación tenga, más complejo, difícil y lento se vuelve administrar su TI.

Por lo tanto, la segunda práctica definitoria para la Etapa 1 es el despliegue en un conjunto estándar de sistemas operativos.

En las grandes empresas, no es raro tener varias aplicaciones que se ejecutan en varios sistemas operativos. Por ejemplo, una aplicación puede ejecutarse en Windows 2012, otra en Windows 2012 R2 y otra en Windows 2016. La eliminación de incluso una de esas variables reduce significativamente la complejidad, además, se vuelve mucho más fácil crear un conjunto de conocimientos compartidos en torno a una pila tecnológica común.

Factores que contribuyen al éxito en la Etapa 1

Encontramos que las siguientes prácticas complementarias tienen un impacto significativo para el éxito en la Etapa 1.

Uso estándar de tecnologías y frameworks: Desde una perspectiva de negocio, la estandarización tiene muchos beneficios: costos de licencia reducidos, capacidad de contratar colaboradores con un conjunto de habilidades específicas; y conocimiento compartido entre equipos, lo que en última instancia conduce a una mayor agilidad y una entrega más rápida de software de mayor calidad.

Configuraciones de aplicaciones en control de versiones: la variación de datos entre los distintos entornos de despliegue requiere almacenar la configuración en control de versiones de manera aislada, para garantizar la protección de las credenciales sensitivas. De igual forma, esto promueve las bases para el despliegue automatizado, el seguimiento a cambios, y revertir despliegues defectuosos fácilmente.

Probar los cambios de infraestructura antes de desplegar en producción: Probar los cambios de infraestructura genera confianza en el sistema para que los equipos puedan ganar autonomía para trabajar sin aprobaciones manuales y también proporciona la base para crear patrones de despliegue reutilizables, lo que no puede hacer a menos que tenga una forma estándar de probar los cambios.

DevOps comienza con la simplificación

Puede sorprender a muchos escuchar que el primer paso en adoptar DevOps no es automatizar, especialmente porque la automatización es un pilar central del movimiento. Las prácticas de automatización de hecho aparecen en una etapa más avanzada (Etapa 4, específicamente), esto se debe a que primero hay que preparar los procesos para que la automatización se diseñe e implemente correctamente.

Las organizaciones que comienzan con la automatización, sin haber pasado por la normalización, estandarización y expansión (Etapas 1-3), no logran el éxito; carecen de una base de colaboración y compartir entre equipos y sus fronteras. Ese intercambio es fundamental para definir los problemas que enfrenta una organización y encontrar soluciones que funcionen para todos los equipos.

Omitir las primeras etapas significa perderse el aprendizaje que tiene. Las primeras etapas también son cuando los equipos que establecen y tienen éxito en las prácticas de DevOps se ganan la confianza de la empresa, lo que puede significar más recursos y permiso para progresar más rápido.

Etapa 2

Estandarizar y reducir la variabilidad

En la **Etapa 1**, normalizamos tecnologías y procesos. Se separaron las configuraciones de las aplicaciones y se colocaron en control de versiones, se adaptó un proceso coherente para las pruebas de infraestructura y un patrón para compartir en base al código fuente.

En la **Etapa 2**, las organizaciones trabajan para estandarizar aún más y reducir la variabilidad, un objetivo que debe prevalecer en cada etapa de la evolución de DevOps.

Toda organización tiene variación, que puede provenir de varias causas diferentes:

- **Adopción de nuevas tecnologías para reemplazar muchas funciones antiguas; sin embargo, estas funciones antiguas nunca se eliminan**
- **Soluciones en casa que no siguen ningún estándar común de la industria y carecen de interfaces comunes**
- **Una propagación de herramientas que se superponen y no han sido justificadas**
- **Fusiones y adquisiciones**

En la Etapa 2, la reutilización de tecnologías y patrones se vuelve importante. Esto impulsa a Dev y Ops a colaborar y tomar decisiones arquitectónicas que afectan el despliegue y capacidad de prueba de las aplicaciones.

¡Cuidado!
Un antipatrón a tener en cuenta en esta etapa es que cada equipo se normalice según sus propios estándares. Esto conducirá a un m ayor grado de variación global y es exactamente la dirección equivocada.

Acumulación de la deuda técnica

Una de las barreras para adoptar DevOps es la gran complejidad de la organización. A medida que las empresas crecen con el tiempo, inevitablemente agregan nuevas aplicaciones y servicios, adoptan nuevas pilas de tecnología y aún tienen que lidiar con aplicaciones y sistemas antiguos o heredados. La deuda técnica se acumula.

Debido al incremento de la complejidad, los sistemas frágiles y los procesos variables, los equipos pasan la mayor parte de su tiempo siendo reactivos, en vez de innovar. El objetivo de esta etapa no es para adoptar una nueva pila tecnológica y rediseñar todo (otra vez). En cambio, necesita estandarizar en tecnologías probadas, optimizar para el 80% de los casos y los casos de uso globales.

El principal beneficio en esta etapa es reducir las variables y, por lo tanto, la complejidad, ganando tiempo para futuras inversiones en colaboración, automatización, compartir y métricas en etapas posteriores.

Aplicación del método científico a la reducción de variables en el software.
El número de variables en cualquier proceso o sistema es directamente proporcional a su complejidad. Con menos variables en juego, es más fácil ejecutar un proceso. También se puede aislarlas, modificarlas y medir el impacto de cada cambio.

Prácticas que definen la Etapa 2

Para lograr estandarizar y reducir (aún más) la variabilidad en esta etapa, nos basaremos en las siguientes prácticas:

Uso estándar de tecnologías y frameworks

Despliegues en un único sistema operativo estándar

Exploremos más a detalle el alcance de estas prácticas.

Uso estándar de tecnologías y frameworks

Tomando la perspectiva de un equipo encargado de la entrega de un producto, sin importar cómo está compuesto, el equipo como necesidad primaria necesita desplegar servicios (o funcionalidades de un servicio). De esta necesidad surgen las preguntas:

- **¿Cada servicio se basa en la misma arquitectura?**
- **¿Todos usan la misma cola de mensajes?**
- **¿Todos usan los mismos componentes y patrones?**

Cuando la respuesta a cualquiera de estas preguntas es "No", la complejidad entra en el sistema y aumenta la carga de mantenimiento.

Cuando se estandarizan patrones y componentes, el equipo ya no tiene que volver a aprender continuamente cómo funcionan, escalan, fallan, recuperan y actualizan las diferentes tecnologías. El tiempo recuperado se puede utilizar para aumentar la velocidad o para desarrollar funcionalidades que diferencien la aplicación o el servicio, los cuales pueden ayudar a proporcionar una ventaja competitiva para toda la organización.

La necesidad de normalización y estándares no significa que los equipos no deban innovar. Para probar algo debe haber una barrera menor y poca resistencia a la experimentación. La barrera debería aumentar significativamente cuando se trata de introducir una nueva pieza de tecnología en un ciclo de vida de producción.

Los beneficios clave de estandarizar los patrones de un equipo y tecnologías son:

- **Mayor velocidad de entrega**
- **Más flexibilidad para que el personal de desarrollo trabaje en diferentes aplicaciones, servicios o componentes**
- **Superficie reducida para vulnerabilidades de seguridad**
- **Menos piezas móviles para mantener, actualizar y aprender**

Despliegues en un único sistema operativo estándar

Cuando existen varios equipos de desarrollo de software, se puede dar la situación en la que cada equipo de manera aislada intente normalizar y estandarizar bajo sus propias condiciones y contexto. A pesar de que esto es justamente normalizar y estandarizar, no es lo que se debe perseguir. ¿Qué pasa con el equipo de operaciones en esta situación? Si Operaciones admite varios equipos, es posible que tengan varias pilas normalizadas que gestionar. Aquí es donde vemos la importancia de optimizaciones más globales.

Permitir que las operaciones seleccionen estándares para el sistema operativo y las versiones, monitorear las interfaces y los sistemas de despliegue, puede eliminar varios debates para los equipos de software, dándoles más tiempo para concentrarse en su misión principal.

Las organizaciones pueden avanzar más rápido cuando el estándar es un solo sistema operativo o un pequeño conjunto de sistemas operativos. Ahorra tiempo en parchear, ajustar, actualizar y solucionar problemas.

Un camino común a seguir: eliminar primero cualquier sistema operativo perdido en su flota. Si tiene cinco sistemas operativos, reduzca a dos. A continuación, normalice sus recursos informáticos que ejecutan el mismo sistema operativo. Asegúrese de que tengan los mismos conjuntos de parches, el mismo nivel de actualización, el mismo BIOS/firmware (si corresponde) y pronto. Si no haces esto, todas esas variables harán que sea mucho más complejo para solucionar problemas y realizar el mantenimiento. Una vez normalizas, tendrás más tiempo y atención disponible para nuevas mejoras y la base adecuada sobre la que construirlas.

Incluso si no puede llegar a un solo sistema operativo, menos es más.

Más allá de la estandarización del sistema operativo está el resto de la pila de tecnología. Los propietarios y electores de las tecnologías pueden variar. Estandarizar a través de muchos equipos sobre opciones tecnológicas como sistemas de bases de datos, colas de mensajes, utilidades de agregación de registros y colección e instrumentación de métricas, permiten que cualquier lección aprendida en el soporte y mantenimiento de esas herramientas pueda ser compartida a otras aplicaciones y equipos.

Normalizar en estos espacios es un reto. A menudo, un equipo de desarrollo gravitará hacia una nueva tecnología, o hacia algo que se especialice en un problema que tenga, sin tener en cuenta la carga de las operaciones y el costo de cuidar otra pieza de tecnología.

Factores que contribuyen al éxito en la Etapa 2

Hay dos prácticas que tienen un impacto significativo en el éxito de la Etapa 2:

- Rediseñar las aplicaciones según las necesidades del negocio: En la Etapa 2 es recomendable aplicar cambios de arquitectura que mejoren la adaptación de las aplicaciones las futuras prácticas y hagan sentido para los objetivos que persigue el negocio. El objetivo principal es admitir la estandarización y alinearse con los objetivos: mayor velocidad y facilidad de mantenimiento.
- Colocar las configuraciones del sistema en control de versiones: Los equipos que usan control de versiones tienen un rendimiento de TI más alto que los que no lo hacen. Es una gran mejora con respecto a los scripts que residen en las estaciones de trabajo de las personas, lo que brinda una serie de ventajas:
  - **Los cambios son visibles a lo largo del tiempo, se puede observar su evolución**
  - **Cualquier persona con acceso puede auditar los cambios**
  - **Existencia de una copia de seguridad histórica automática**

  Esta práctica es uno de los primeros pasos para adoptar prácticas de desarrollo de software en infraestructura. Esto, a su vez, es clave para la entrega de infraestructura automatizada y un elemento fundamental para la infraestructura como código.

Etapa 3

Expandir las prácticas de DevOps

Las Etapas 1 y 2 reducen la complejidad general de la pila tecnológica para que los equipos puedan lograr resultados más repetibles con una variación limitada. La Etapa 3 se trata de la expansión de las prácticas de DevOps, más allá de los equipos Dev y Ops, a los grupos de equipos en TI y entrega de servicio.

A medida que aumenta la colaboración y la organización se enfoca en mejorar la gestión de servicios, despliegue, la reducción de los tiempos de espera y la minimización de las aprobaciones, estos esfuerzos tocan áreas más allá de los departamentos de tecnología.

Compartir herramientas, aplicaciones y servicios mejorados, así como el conocimiento adquirido, con otras áreas funcionales del negocio se vuelve clave para expandir y escalar DevOps en toda la organización.

Prácticas que definen la Etapa 3

Expandir las prácticas de DevOps es una ardua tarea, para tener éxito en esta etapa nos enfocaremos en las siguientes prácticas:

- **No existe aprobación manual externa al equipo**
- **Reutilización de los patrones de despliegue para aplicaciones y servicios**
- **Los cambios de infraestructura son probados antes de su despliegue en producción**

Exploremos más a detalle el alcance de estas prácticas.

No existe aprobación manual externa al equipo

Tener procesos de aprobación de cambios externos tiene un impacto despreciable en la estabilidad, pero un efecto perjudicial en la agilidad. A pesar de esta evidencia, con frecuencia la autoridad para tomar decisiones se elimina de las personas que tienen la información relevante y están haciendo el trabajo real.

Cuando alguien puede hacer el trabajo con transferencias, aprobaciones y tiempo de espera mínimos, es más feliz y productivo. La Etapa 3 es donde la burocracia debe reducirse y los procesos deben redefinirse y actualizarse para reflejar la confianza mutua que se gana a través de la adaptación de DevOps.

El principal impulsor del proceso burocrático es normalmente la comunicación y la difusión de posibles impactos y problemas. Si se tienen en cuenta esas exigencias al acortar y simplificar el proceso de control de cambios, las personas que desean mejorar la funcionalidad técnica pueden comenzar a revertir su visión del control de cambios como un obstáculo.

¡Cuidado!
A veces, las personas pueden hacer mucho trabajo sin aprobación externa al equipo simplemente porque nadie más tiene conocimiento al respecto. Si bien eso logra que "las personas puedan hacer el trabajo sin aprobación manual fuera de el equipo", no es el mejor camino a seguir.

Reutilización de los patrones de despliegue para aplicaciones y servicios

La reutilización de patrones de despliegue puede significar simplemente que, de tener dos proyectos de software, ambos se despliegan de la misma manera, ya sea para los entornos de dev, test, stage o production. Alguien que despliegue la aplicación A debería poder desplegar la aplicación B sin mucha documentación y acompañamiento.

Algunas organizaciones comienzan estandarizando los puntos de entrada para el despliegue; por ejemplo, para desplegar cualquier aplicación, bastaría escribir ./deploy <entorno>. El resto del proceso de despliegue puede variar de una aplicación a otra, pero al menos se tiene la misma invocación. Eso es un buen comienzo.

El siguiente paso es utilizar las mismas herramientas para sus despliegues. Por ejemplo, todos los despliegues pueden ejecutarse a través de la integración continua (CI), con el sistema de CI ejecutando un conjunto de jobs que dan como resultado un despliegue completo después de terminar.

En organizaciones donde los patrones de despliegue realmente se dominan, múltiples aplicaciones usan los mismos pipelines y jobs para el despliegue; solo el nombre de la aplicación y posiblemente algunos otros parámetros se envían como configuración.

Los cambios de infraestructura son probados antes de su despliegue en producción

Una práctica asociada con la Etapa 3 es probar los cambios de infraestructura antes de desplegarlos en producción. Para tener éxito en la expansión de la iniciativa DevOps, debe demostrar la capacidad de realizar cambios predecibles y confiables.

Es normal pensar que las pruebas de infraestructura deben estar automatizadas, tomando en cuenta lo que se aprende sobre la integración continua y con el enfoque de infraestructura como código. Si bien la automatización es

más confiable y generalmente más rápida, es bueno recordar que lo que realmente importa es la validación y que puede probar los cambios de infraestructura manualmente.

¿Por qué es esto? Debido a que los cambios de infraestructura pueden variar ampliamente, y aunque algunos se prestan a la automatización con una cantidad razonable de esfuerzo, otros cambios son demasiado poco frecuentes o costosos para validarlos de manera automatizada. Así que no es ideal encerrarse demasiado en el método, solo se debe asegurar validar los cambios de infraestructura antes de un despliegue a producción.

Si se adoptan varios entornos de prueba que repliquen la configuración de producción, las pruebas de infraestructura pueden ocurrir de manera natural en el proceso de despliegue de estos ambientes de aplicación.

Factores que contribuyen al éxito en la Etapa 3

Hay cuatro (4) prácticas que tienen un impacto significativo en el éxito de la Etapa 3:

- Se realizan cambios sin tiempos de espera significativos: Las organizaciones deben trabajar para reducir los tiempos de espera de las aprobaciones, esto hace que sea más difícil ser ágil y van en contra del principio de DevOps de empoderar a las personas y los equipos.

  Trabajar para lograr cambios y despliegues sin tiempos de espera no se trata de obtener un permiso o pase para eludir el proceso organizacional. Se trata de revisar por qué existe un proceso para que pueda simplificarlo, normalizarlo y optimizarlo. Cuando los procesos son más simples y consistentes, también son más fáciles de automatizar.

- Los cambios se pueden desplegar durante horario de trabajo: Una vez que se han analizado los tiempos de espera para reducirlos, puede pasar a realizar tareas de mantenimiento del servicio y cambios durante el horario de trabajo.

  Algunas organizaciones logran esto haciendo uso de estrategias de despliegues como Canary, Azul/Verde, entre otros. Estos patrones permiten un plan de reversión relativamente fácil si un cambio sale mal. Pero, llegar al punto en el que puede realizar cambios durante horario de trabajo requiere algo de preparación. Esta preparación la podemos dividir en dos pasos esenciales:

  I. **Primero, debe definir qué significa el horario de trabajo para su organización. Si está siempre activo (como un servicio web), los clientes internos y externos pueden esperar que su servicio esté disponible todo el tiempo.**

  II. **En segundo lugar, debe demostrar éxito en la realización de cambios de manera confiable para que los socios comerciales y las partes interesadas de su servicio confíen en sus habilidades. Necesita que le crean cuando dice que los cambios no tendrán ningún impacto en el rendimiento ni en los clientes.**

- Las revisiones posteriores a incidentes ocurren y se comparten los resultados: Las revisiones posteriores son una mirada irreprochable a lo que sucedió durante un incidente, cómo sucedió y qué mejoras se podrían hacer para acortar la duración del incidente, mejorar la comprensión de los sistemas detrás del incidente y evitar que vuelva a suceder.

  Las revisiones posteriores a incidentes surgen tanto del pilar del compartir de CAMS como de los principios de DevOps; se diseñan para dar un enfoque hacia la compresión y la mejora continua, y evitan caer en los enfoques tradicionales de revisión post-acción.

  Las mejoras de una revisión posterior a incidentes bien realizada pueden incluir:

  - **revisar y simplificar los procesos;**
  - **actualizar los patrones de comunicación;**
  - **y trabajar desde una posición de empatía con otras partes interesadas de la aplicación o servicio.**

  Una vez que se realiza una revisión posterior al incidente, es vital compartir los resultados. Las personas que no estuvieron directamente involucradas pueden aprender algo. Pueden detectar una falla en un proceso adyacente, o simplemente sentir curiosidad por lo que sucedió cuando no pudieron acceder a su sitio durante horas. Cuanto más comparta, más colaboración y confianza fomentará.

- Los equipos utilizan integración continua: Hemos visto el uso de CI como un indicador principal de si un equipo tendrá un alto rendimiento o no. CI es imprescindible en el espacio DevOps, justo después de que el control de versiones se vuelve omnipresente.

  Los sistemas de CI y flujos de pipelines pueden variar enormemente según:

  - **los tipos de software,**
  - **la estructura de los jobs,**
  - **las herramientas y**
  - **quién consume el flujo de trabajo.**

Las cosas importantes para optimizar son el tiempo del ciclo de retroalimentación y la corrección. Cuando los tiempos del ciclo de retroalimentación son cortos, pueden ocurrir más iteraciones y, por lo tanto, la calidad mejora.

La corrección también es importante, por lo que los sistemas de CI requieren mantenimiento, ajustes y mejoras con el tiempo. Por ejemplo, si agrega un nuevo sistema operativo o navegador a su matriz de soporte, todos los trabajos relevantes deberían poder recogerlo.

Etapa 4

Automatizar la entrega de infraestructura

La razón por la que esta práctica aparece ahora y no antes, es porque las etapas anteriores facilitan el éxito que se puede obtener con la automatización en general.

Infraestructura ágil

Los equipos de infraestructura en esta etapa comienzan a adoptar prácticas de desarrollo ágiles, como el uso del control de versiones tanto para la configuración del sistema como para la configuración de las aplicaciones. Los equipos en esta etapa también automatizan las configuraciones de políticas de seguridad dentro de su esfera de influencia.

Prácticas que definen la Etapa 4

Las prácticas definitorias para la Etapa 4 son:

Configuraciones del sistema automatizadas

Aprovisionamiento automatizado

Las configuraciones de la aplicación están en el control de versiones

Los equipos de infraestructura utilizan control de versiones

Exploremos más a detalle el alcance de estas prácticas.

Configuraciones del sistema automatizadas

Cuando se habla del pilar de automatización en el modelo CAMS, generalmente se refiere justo a esta práctica: automatizar las configuraciones del sistema y mantenerlas en el control de versiones.

Entre algunos de los beneficios que podemos mencionar, a parte de la aceleración de los cambios, se encuentran:

Velocidad total: Las tareas automatizadas deberían ser completadas más rápido que las tareas completadas manualmente.

Consistencia: Las tareas automatizadas siguen un proceso establecido y, por lo tanto, producen resultados predecibles.

Comportamiento documentado: Las tareas ahora tienen una forma definida en la que se supone que funcionan, por lo que son más fáciles de solucionar.

Al comenzar con elementos de [ROI] más altos, se paga la inversión en automatización de manera efectiva de inmediato. Este tiempo se puede dedicar a automatizar más cosas, simplificar procesos o mejorar otros servicios integrados en la infraestructura.

Cuando se empieza a automatizar, se deben priorizar los elementos que se gestionan con más frecuencia en la amplia variedad de componentes de infraestructura. Esto tendrá un gran impacto: liberará tiempo de manera significativa por lo que permitirá trabajar en automatizaciones más complejas.

Aprovisionamiento automatizado

El aprovisionamiento automatizado es otra práctica definitoria en la Etapa 4. Cuando se combina con configuraciones de sistemas automatizados, se obtiene la base de una infraestructura de autoservicio (cubierta en la Etapa 5).

El aprovisionamiento puede ser la creación automática de un recurso de casi cualquier tipo. La mayoría de las veces, los equipos usan la palabra cuando hablan de:

- **instancias de sistemas operativos,**
- **conectividad de red,**
- **almacenamiento y**
- **cuentas.**

Sin embargo, algunos equipos llevan el aprovisionamiento automatizado mucho más allá, con enlaces a servicios de:

- **DNS,**
- **[CDN (Content Delivery Network)][CDN],**
- **balanceadores de carga,**
- **bases de datos, y más.**

Al igual que con las configuraciones del sistema, es mejor comenzar con el elemento solicitado con más frecuencia; obtenga algunas victorias, consistencia y ahorro de tiempo; y luego pasar a la siguiente solicitud más frecuente.

En este punto, el aprovisionamiento podría realizarse con un marco, o incluso con un conjunto de scripts y utilidades compartidos almacenados en el control de versiones. La clave es que el equipo pueda subir de nivel y realizar el aprovisionamiento de forma automatizada. Un criterio de aceptación clave para un buen aprovisionamiento automatizado es

Al igual que con la mayoría de los pasos en la evolución de DevOps, es bueno perseguir y elegir tareas que ganen la confianza, incluso la gratitud, de los demás, tanto dentro como fuera de su equipo.

que el cliente no puede saber a quién se le asignaron las tareas de aprovisionamiento (lo que implica que las personas individuales del equipo de infraestructura no realizaron ajustes especiales).

Las configuraciones de la aplicación están en el control de versiones

A estas alturas, el código fuente de su aplicación está en control de versiones. Excelente. ¿Qué pasa con la configuración de su aplicación? Muchas aplicaciones, ya sean listas para usar o desarrolladas internamente, se construyen utilizando algunos patrones de una aplicación de [12 factores][12-factor application]. Incluso si una aplicación no sigue los 12, las configuraciones normalmente se externalizan, no se codifican en la aplicación.

La configuración que hace que la implementación de su aplicación funcione en su entorno es fundamental. Atrás quedaron los días en que un administrador iniciaba sesión directamente en un sistema y luego editaba manualmente un archivo de configuración.

Las configuraciones de la aplicación deben ser:

- **versionadas,**
- **auditables,**
- **contener historial e,**
- **idealmente, las razones por las que se han cambiado.**

Separar las configuraciones de su aplicación del código fuente permite desplegar y validar el mismo código fuente y artefacto en múltiples entornos, con el único cambio en la configuración.

Esta separación se vuelve primordial cuando desea aprovisionar entornos de desarrollo individuales o avanzar hacia el autoservicio. No es eficiente volver a compilar una aplicación para todos y cada uno de los usuarios, o codificar sus parámetros en ella. Por lo tanto, separar el código de los datos de configuración permite implementaciones, actualizaciones y validaciones más rápidas.

Los equipos de infraestructura utilizan control de versiones

Uno de los cambios importantes que ocurren en la Etapa 4 es que los equipos de infraestructura adoptan buenas prácticas de desarrollo, como el uso del control de versiones.

Como se señaló anteriormente, el uso del control de versiones para todos los artefactos de producción está altamente correlacionado con el rendimiento de TI. Es el primer paso para la entrega continua de su código de infraestructura. El uso del control de versiones facilita la recreación de entornos para pruebas y solución de problemas, lo que aumenta el rendimiento tanto para desarrollo como para operaciones. También reduce el tiempo de recuperación si se identifica un error en la producción. Puede volver a desplegar rápidamente el último buen estado o solucionar el problema y avanzar, todo con capacidades de historial y auditoría.

Factores que contribuyen al éxito en la Etapa 4

Para la Etapa 4, contamos con una práctica que contribuye al éxito:

- Automatizar las configuraciones de políticas de seguridad: Más allá de tener políticas de seguridad, las organizaciones a menudo tienen restricciones externas que requieren la demostración y el cumplimiento de las políticas de seguridad a través de controles medidos.

Esas fuerzas externas pueden ser:

- **Un comité de auditoría,**
- **Estándares de la industria de tarjetas de pago (PCI),**
- **NIST,**
- **Pautas generales de protección de datos (GDPR) y**
- **una miríada de otros estándares regulatorios.**

En este punto de la evolución de DevOps, la mayor parte de la automatización de políticas de seguridad ocurre a nivel de equipo, con el objetivo principal de garantizar que la interacción con los auditores se mantenga al mínimo. Se hace de esta manera por razones prácticas, ya que el nivel de equipo es donde las transferencias son mínimas y el [ROI] se obtiene de inmediato.

La mejor manera de adherirse a la política de seguridad es saber si se cumple y reparar los sistemas cuando no cumplen. Las organizaciones que evolucionan en su viaje de DevOps hacen precisamente eso.

A medida que la automatización de políticas de seguridad madura un poco, surge el uso de sistemas de gestión de configuración. La gestión de la configuración permite que

A medida que la organización mejore y evolucione, comenzará a resolver el problema de las políticas de seguridad de manera más amplia.

se apliquen las políticas sobre la convergencia del sistema y que los informes se gestionen de forma estándar. Colocar la política de seguridad en la gestión de la configuración de la infraestructura actúa como una función de normalización para el equipo, lo que significa que cualquier miembro del equipo puede actualizar o mejorar la aplicación de la política a través del código base.

Desde la perspectiva de un equipo de desarrollo de aplicaciones, la infraestructura en la que se basa debe cumplir con la política de seguridad. Una herramienta de administración de la configuración hará cumplir constantemente las políticas correctas debajo de las aplicaciones. Sin embargo, esto no significa que los equipos de entrega de aplicaciones no tengan nada que hacer. Algunos equipos pueden ejecutar análisis estáticos en el código a través de sus canalizaciones de integración continua. Algunos equipos también usarán herramientas externas para escanear sus aplicaciones en busca de vulnerabilidades, como el top **10 de OWASP** a través de herramientas externas.

Etapa 5

Proporcionar capacidades de autoservicio

Para pasar a la **Etapa 5**, una organización debe tener varios departamentos comprometidos con brindar capacidades de TI como un servicio al negocio, en lugar de tratar a TI como un centro de costos que ejecuta órdenes de trabajo.

Estos departamentos incluyen:

- ∞ **desarrollo de software**
- ○ **operaciones**
- □ **seguridad**
- △ **ITSM**
- \+ **otras áreas funcionales**

En esta última etapa de la evolución de DevOps, los beneficios para la organización se multiplican enormemente a medida que se acelera la colaboración exitosa.

Estas ganancias se ven en varias áreas distintas:

- La arquitectura de la aplicación comienza a evolucionar hacia el soporte de la migración a la nube, la adopción de contenedores y la proliferación de microservicios.
- La automatización de políticas de seguridad pasa de atender las necesidades de un equipo a convertirse en la línea de base sobre cómo se miden la seguridad y el cumplimiento en todo un departamento, o incluso en toda la organización.
- El aprovisionamiento automatizado avanza hacia el aprovisionamiento de entornos completos para desarrolladores, ingenieros de calidad y cualquier otro personal técnico.

Una vez que comienza a tener éxito en múltiples fronteras funcionales, los pilares de DevOps (cultura, automatización, medición y compartir) se vuelven más omnipresentes en toda la organización.

Prácticas que define la Etapa 5

Las tres (3) prácticas definitorias para la Etapa 5 son:

- Las respuestas a incidentes están automatizadas
- Los recursos están disponibles a través del autoservicio
- Los equipos de seguridad participan en el diseño y la implementación de la tecnología

Las respuestas a incidentes están automatizadas

Responder manualmente a incidentes críticos es costoso de varias maneras:

- en términos de atención y enfoque de ingeniería y,
- en términos del tiempo que lleva detectar, identificar y remediar el incidente.

Particularmente cuando se trata de intrusiones o malware, una respuesta puede ser muy costosa si no soluciona el problema por completo. La falta de una sola máquina infectada es todo lo que se necesita para que toda la respuesta sea inútil.

Todo esto significa que se puede obtener una gran cantidad de valor al automatizar la respuesta a incidentes. La automatización elimina las distracciones innecesarias, mejora el tiempo de remediación al reducir las transferencias y garantiza que sus procesos de remediación se apliquen de manera consistente.

La automatización completa de su sistema de respuesta a incidentes es una tarea abrumadora, pero no necesita automatizar para cada tipo de incidente. En cambio, piense en su automatización como si estuviera allí para aumentar el juicio humano. Céntrese en los procesos y sistemas que le permiten identificar problemas, así como en los que implementa al responder. Facilite que sus operadores obtengan los datos que necesitan para formarse un juicio y, una vez que lo hayan hecho, automatice los procesos de respuesta, como agregar una IP maliciosa a todos sus firewalls en toda su infraestructura; recopilación de datos para análisis forenses posteriores; o aislar completamente una máquina infectada.

Para aquellos en entornos más pequeños, puede que no sea inmediatamente obvio por qué las respuestas automatizadas a incidentes no entran en juego hasta la Etapa 5. Se debe a importantes barreras organizacionales y de procesos en las empresas que a menudo impiden que las personas de respuesta a incidentes logren una remediación completa.

- **Falta de acceso a métricas y registros;**
- **tener que presentar tickets para que otros validen los cambios en las reglas del cortafuegos;**
- **la necesidad de firmas de los propietarios del servicio;**
- **la incapacidad de impulsar los cambios finales hasta la producción.**

Todas estas barreras para la retroalimentación rápida y los ciclos de acción deben eliminarse para agregar automatización a sus respuestas a incidentes. Muchas empresas no han llegado a este punto.

Es necesario contar con algunas capacidades técnicas fundamentales para brindar respuestas automatizadas a incidentes, pero es igualmente importante tener una relación de colaboración con su equipo de seguridad. Incorpore al equipo de seguridad al ciclo de vida de desarrollo de manera temprana y comience poco a poco automatizando en colaboración la respuesta a un tipo específico de incidente que cruza los límites funcionales.

Los recursos están disponibles a través del autoservicio

Los excelentes sistemas de autoservicio son un facilitador increíble en una empresa. Cuanto más empodere a las personas para que trabajen a su propio ritmo sin tener que esperar (a que se aprueben los tickets, se obtengan las claves de licencia, se actualicen las configuraciones de red o se apliquen las configuraciones requeridas), menos frustración experimentarán las personas, más fácil será estandarizar configuraciones, y el trabajo será más predecible.

Tener que cambiar el enfoque mientras esperas a que alguien más mate el progreso, sin mencionar el entusiasmo por la tarea en cuestión.

Es importante tener en cuenta que puede y debe trabajar hacia el autoservicio mucho antes de esta etapa, y es absolutamente posible ofrecer valor real de forma incremental antes de llegar al punto de un catálogo de autoservicio completo. Los equipos deben crear sistemas de autoservicio para ellos y luego para sus equipos adyacentes, y luego expandirse hacia el exterior a través de la organización.

Los equipos de seguridad participan en el diseño y la implementación de la tecnología

Como discutimos en la Etapa 0: construir la base, el cambio a la izquierda se trata de incorporar más equipos al proceso de desarrollo y entrega, por ejemplo, calidad, seguridad, base de datos, auditoría y redes. La mayoría de los equipos comienzan el cambio hacia la izquierda abordando el dolor de despliegue, que es el límite funcional entre Dev y Ops. Esto representa un paso a la izquierda, mientras que involucrar a otros equipos, como calidad y seguridad, significa cambiar el enfoque varios pasos más a la izquierda, mucho antes del despliegue. Por lo tanto, tiene sentido que los equipos de seguridad se involucren más adelante en la evolución de DevOps, después de que se hayan abordado los problemas más agudos.

En el informe [Puppet State of DevOps de 2016], descubrimos que las empresas de alto rendimiento dedican un 50% menos de tiempo a solucionar problemas de seguridad que las de bajo rendimiento. Esto se debe a que incorporan seguridad en el ciclo de entrega de software en lugar de actualizar la seguridad al final.

En la Etapa 5, los equipos de seguridad deben involucrarse desde el principio en el proceso de desarrollo de software al:

- Llevar a cabo una revisión de seguridad para todas las características principales y garantizar que el proceso de revisión de seguridad no ralentice el desarrollo.
- Integrar la seguridad de la información en el trabajo diario de todo el ciclo de vida de entrega de software. Esto incluye proporcionar información durante el diseño de la aplicación, asistir a demostraciones de software y proporcionar comentarios durante estas demostraciones.
- Probar los requisitos de seguridad como parte del proceso de prueba automatizado.

Factores que contribuyen al éxito en la Etapa 5

El valor que deben proporcionar sus equipos de operaciones y seguridad es su experiencia obstinada, y cuanto más pueda integrar esta experiencia en el software implementado, mejores serán sus resultados. Esto no significa que ignore las necesidades de los usuarios que consumen el software: debe comprender lo que intentan lograr, tener empatía por el entorno en el que trabajan y equilibrarlo con los requisitos operativos y de seguridad. En muchos sentidos, esto requiere un cambio hacia una mentalidad de gerente de producto.

Por eso es importante que los equipos comiencen con el autoservicio para su propio uso. Sus propios problemas son los problemas que mejor entienden, por lo que, naturalmente, hay más empatía por el usuario. Servir al cliente que mejor conoce brinda una gran oportunidad para comenzar a aprender cómo desarrollar el autoservicio.

Se vuelve cada vez más difícil entender las necesidades del usuario a medida que el usuario se aleja más y más, organizacional y funcionalmente, del equipo que construye el servicio. Si bien la empatía y la mentalidad de gerente de producto ayudan a cerrar estas brechas, en última instancia, necesitará crear estos servicios en colaboración con los equipos que brindan la funcionalidad adyacente, así como también con los equipos que consumirán su servicio.

Tres (3) prácticas clave tienen un impacto significativo en la Etapa 5:

- Los desarrolladores de aplicaciones implementan entornos de prueba por su cuenta: Cuando los equipos de desarrollo de aplicaciones pueden desplegar entornos de prueba en demanda, son más productivos (porque no tienen que esperar un nuevo entorno, ir y venir a través de tickets, etc.) y la entrega de aplicaciones es más rápida.

  Los equipos de operaciones también se benefician al proporcionar esta capacidad de autoservicio, ya que pueden dedicar más tiempo a optimizar el sistema en lugar de aprovisionar sistemas. Una vez que los desarrolladores pueden implementar entornos de prueba, se vuelve mucho más fácil habilitar despliegues automatizados.

- Automatice las configuraciones de políticas de seguridad: Esta práctica demuestra ser significativa en las etapas evolutivas posteriores de DevOps (consulte Etapa 4: Automatizar la entrega de infraestructura). ¿Por qué? Porque las configuraciones de políticas de seguridad son una de las cosas más difíciles de automatizar.

  A pesar de las dificultades, vale la pena automatizar la configuración de políticas de seguridad. Es mucho más económico prevenir y mitigar los problemas de seguridad en el diseño de la aplicación que reaccionar ante ellos en la producción.

  Es por eso que hay un movimiento de "cambio a la izquierda" en seguridad en este momento. Las consideraciones de seguridad están pasando de ser principalmente preocupaciones operativas en la producción a incorporarse en el diseño y la construcción de aplicaciones. Es un movimiento contrario a la forma tradicional de construir software, donde equipos completamente separados se enfocan en diferentes partes del ciclo de construcción. En este escenario, es muy fácil que cada equipo ignore u olvide las preocupaciones de otros equipos, al menos hasta que ocurra un incidente una vez que la aplicación esté en producción.

  Entonces, al igual que reunir a los equipos de desarrollo y operaciones permite que las consideraciones operativas formen parte del diseño de la aplicación, lo mismo ocurre con los equipos de seguridad y sus áreas de responsabilidad: deben incluirse al principio del ciclo de diseño del software.

- Las métricas de éxito de los proyectos son visibles: Aportando al componente de compartir del modelo CAMS, exponer y dar visibilidad de las métricas de éxito de proyectos, equipos e individuos promueve de manera más ágil el cambio cultural permanente que se quiere lograr.

Cultura
Cómo transformar

Introducción

Todas las organizaciones, sin importar su tamaño, origen, industria o cualquier otro contexto que se pueda mencionar, están constantemente expuestas al cambio. Por lo tanto, se deben hacer algunas preguntas, como:

- **¿Cuál es la dirección de ese cambio?**
- **¿Cuáles son los resultados a nivel del sistema por los cuales está trabajando?**
- **¿Puede la organización descubrir y servir a sus clientes, y así lograr su propósito?**
- **¿El modelo de negocio de la organización y su gestión de personas, proporciona sostenibilidad a largo plazo?**

Cuando parece que las cosas no van a planearse, es común que los líderes implementen un programa de transformación. Sin embargo, estos programas a menudo no logran sus objetivos y consumen grandes cantidades de recursos y capacidad organizativa. Este documento examina cómo ejecutar con éxito una transformación y aborda algunas fuentes comunes de fallas.

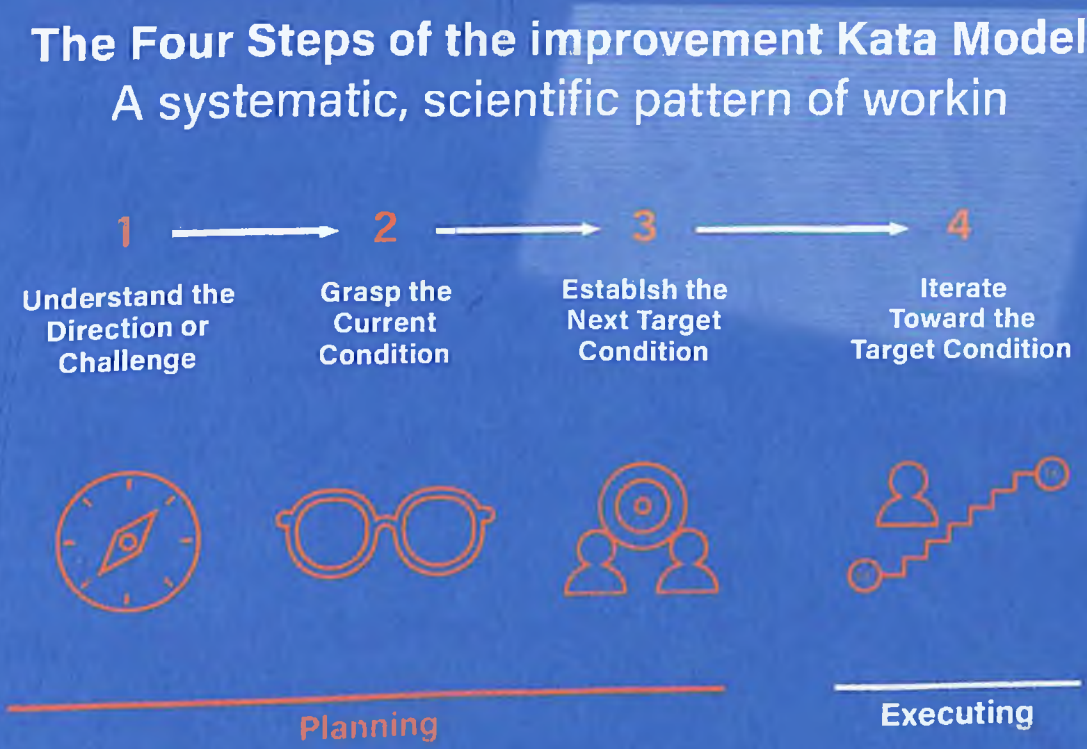
Cómo implementar la transformación

Hay dos ingredientes clave en una transformación efectiva y continua:

- **procesos para ejecutar el cambio organizacional mediante el establecimiento de objetivos y permitiendo la experimentación en equipo,**
- **y mecanismos para difundir las buenas prácticas en toda la organización.**

Establecer objetivos y permitir experimentación

Hay muchos marcos para ejecutar y medir el cambio organizacional, como cuadro de mando integral, objetivos y resultados clave (OKR), el kata de mejora y el kata de entrenamiento. Estos marcos pueden parecer diferentes, pero todos comparten características clave. La dinámica básica se muestra en la siguiente figura, que se basa en el marco de kata de mejora:



Todos los marcos comienzan con una dirección (un "norte verdadero") en el nivel organizacional o de división. Este es un objetivo comercial aspiracional a nivel de sistema establecido por el liderazgo. Podría ser un ideal que no se puede alcanzar, como cero lesiones (la meta escogida por el CEO de Alcoa, Paul O'Neill). O podría ser un objetivo difícil que está a uno o tres años, como un aumento de diez veces en la productividad (el objetivo elegido por Gary Gruver, cuando era director de ingeniería de la división de firmware LaserJet de HP).

El siguiente paso es comprender la condición actual. La verificación rápida de DORA puede ayudarlo a comprender cómo le está yendo en términos de sus capacidades y resultados de desarrollo de software. Otro enfoque de análisis es realizar ejercicios como el mapeo del flujo de valor o la contabilidad de actividades. El punto es entender dónde está la organización en términos medibles.

El tercer paso es establecer objetivos medibles para una fecha futura. Estos objetivos podrían describirse utilizando un formato como los OKR, que comienzan con un objetivo cualitativo y luego especifican resultados clave medibles (condiciones objetivo). Por ejemplo, el CIO de Banca y Mercados Globales de HSBC fijó a cada equipo el objetivo de "duplicar, la mitad y un cuarto cada año: duplicar la frecuencia de los comunicados, la mitad de la cantidad de incidentes de bajo impacto y la cuarta parte de la cantidad de incidentes de alto impacto".

Finalmente, los equipos experimentan con formas de lograr estos objetivos hasta que se alcance la fecha futura, con el apoyo de la gerencia. Los equipos adoptan un enfoque científico para la experimentación, utilizando el método PDCA (Plan-Do-Check-Act), también conocido como el ciclo de Deming.

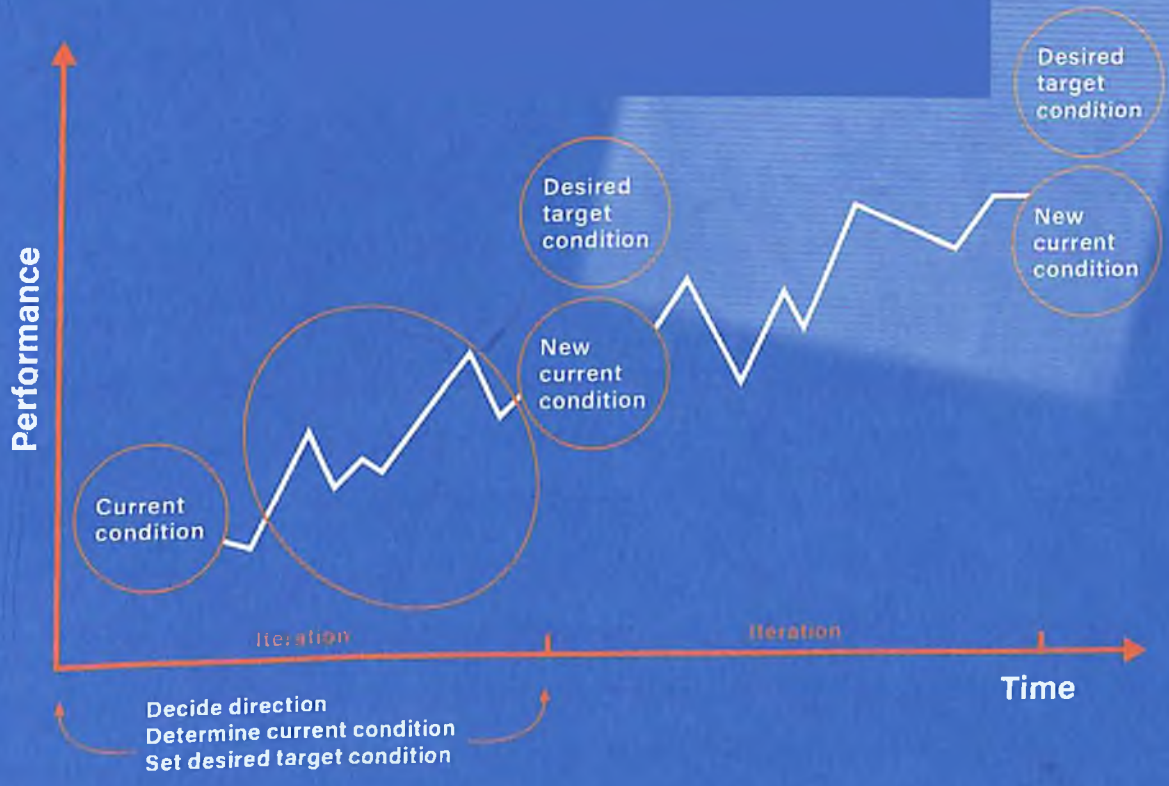
El ciclo consta de los siguientes pasos:

Los equipos deben realizar experimentos a diario para tratar de avanzar hacia las condiciones objetivo o los resultados clave. En el kata de mejora, todos los miembros del equipo deben hacerse las siguientes cinco preguntas todos los días:

- **¿Cuál es la condición objetivo?**
- **¿Cuál es la condición actual?**
- **¿Qué obstáculos crees que te impiden alcanzar la condición objetivo? ¿A cuál te diriges ahora?**
- **¿Cual es tu siguiente paso? ¿Qué resultado esperas?**
- **¿Cuándo se pueden evaluar los resultados para ver qué se puede aprender al dar ese paso?**

Cuando se hayan capturado los resultados y se hayan establecido nuevos objetivos, repita el proceso.

Debido a que el proceso se realiza en condiciones de incertidumbre, no siempre está claro cómo se lograrán los resultados. Por lo tanto, el progreso suele ser no lineal, como se muestra en el siguiente diagrama:



En las reuniones de planificación, los participantes revisan las condiciones objetivo o los resultados clave que se establecieron en la última reunión de planificación. Luego establecen nuevos objetivos para la próxima iteración. En las reuniones de revisión, los participantes observan qué tan bien los equipos están logrando los objetivos de la iteración y discuten los obstáculos y cómo se abordarán.

Algunos puntos importantes sobre este patrón son los siguientes:

- El propio equipo debe establecer las condiciones objetivo u OKR de un equipo. Si se establecen de forma descendente, los equipos no tendrán interés en el resultado y, por lo tanto, no invertirán tanto en lograrlos. En cambio, el equipo podría "jugar con ellos", es decir, manipular el resultado para alcanzar la meta artificialmente.
- Es aceptable no lograr las metas; algunos de los objetivos son objetivos ambiciosos, lo que significa que están diseñados a propósito para ser desafiantes. Los equipos deben esperar lograr alrededor del 80% de las metas. Es común que cuando se comienza con la transformación cultural no se alcance ninguna de las metas especificadas. Si esto sucede, el equipo debe establecer un objetivo único para la próxima iteración y dedicar todo a lograrlo.
- Muchos objetivos y medidas cambiarán de una iteración a otra a medida que cambien los objetivos del equipo y las condiciones actuales, y a medida que aprendan a través del trabajo hacia sus objetivos. No pierda demasiado tiempo tratando de establecer los objetivos perfectos: concéntrese en ejecutar el proceso para que pueda comenzar a aprender.
- Es importante que los equipos tengan la autonomía, la capacidad, los recursos y el apoyo de gestión y liderazgo necesarios para realizar el trabajo de mejora. Los equipos no deben permitir que el trabajo de entrega normal desplace el trabajo de mejora, porque el trabajo de mejora es lo que ayudará a corregir las ineficiencias que hacen que la entrega de productos y servicios sea tan lenta.

Construir estructuras de comunidades para difundir el conocimiento

El análisis muestra que las empresas de alto desempeño favorecen las estrategias que crean estructuras comunitarias tanto en los niveles bajos como altos de la organización, lo que probablemente los hace más sostenibles y resistentes a las reorganizaciones y los cambios de productos. Las dos principales estrategias empleadas son las comunidades de práctica y de base, seguidas de la prueba de concepto como plantilla (un patrón en el que la prueba de concepto se reproduce en otras partes de la organización) y la prueba de concepto como semilla.

Los de bajo rendimiento tienden a favorecer los centros de formación y los centros de excelencia: estrategias que crean más silos y experiencia aislada. También intentan pruebas de concepto, pero generalmente se estancan y no tienen éxito. ¿Por qué es posible que estas estrategias no logren generar un cambio efectivo?

Al centralizar la experiencia en un grupo, los centros de excelencia crean varios problemas. Primero, el CoE ahora es un cuello de botella para la experiencia relevante para la organización y esto no puede escalar a medida que crece la demanda de experiencia en la organización. En segundo lugar, establece un grupo exclusivo de "expertos" en la organización, en contraste con un grupo inclusivo de pares que pueden seguir aprendiendo y creciendo juntos. Esta exclusividad puede socavar culturas organizacionales saludables. Finalmente, los expertos se retiran de hacer el trabajo. Pueden hacer recomendaciones o establecer "mejores prácticas" genéricas, pero el camino desde el aprendizaje genérico hasta la implementación del trabajo real se deja en manos de los alumnos. Por ejemplo, los expertos crearán un taller sobre cómo contenerizar una aplicación, pero rara vez o nunca realmente contienen aplicaciones. Esta desconexión entre la teoría y la práctica eventualmente amenazará su experiencia.

Si bien algunos ven el éxito en los centros de capacitación, requieren recursos y programas dedicados para ejecutar tanto el programa original como el aprendizaje sostenido. Muchas empresas han reservado recursos increíbles para que sus programas de capacitación sean efectivos: tienen edificios completos dedicados a un entorno creativo separado y personal dedicado a crear materiales de capacitación y evaluar el progreso. Entonces se necesitan recursos adicionales para asegurar que el aprendizaje se sostenga y se propague en toda la organización. La organización debe brindar apoyo a los equipos que asistieron al centro de capacitación, para ayudar a garantizar que sus habilidades y hábitos continúen en sus entornos de trabajo habituales y que no se reanuden los viejos patrones de trabajo. Si estos recursos no están en su lugar, las organizaciones corren el riesgo de que todas sus inversiones se

desperdicien. En lugar de un centro donde los equipos van a aprender nuevas tecnologías y procesos para difundirlos al resto de la organización, los nuevos hábitos se quedan en el centro, creando otro silo, aunque sea temporal. También existen limitaciones similares a las del CoE: si solo el personal del centro de capacitación (u otros "expertos independientes") están creando talleres y materiales de capacitación, ¿qué sucede si en realidad nunca hacen el trabajo?

Se informaron comúnmente mashups (40% de las personas que respondieron a la encuesta de 2019 utilizaron esta estrategia), pero carecen de fondos y recursos suficientes en cualquier inversión en particular. Sin una estrategia para guiar una transformación tecnológica, las organizaciones a menudo cometerán el error de cubrir sus apuestas y sufrirán la muerte por iniciativa: identificar iniciativas en demasiadas áreas, lo que en última instancia conduce a la escasez de recursos para el trabajo importante y los condena a todos al fracaso. En su lugar, es mejor seleccionar algunas iniciativas y dedicar recursos continuos para garantizar su éxito (tiempo, dinero y patrocinio de ejecutivos y practicantes campeones). En contraste con los mashups, muy pocas personas reportan el uso de una estrategia big bang, aunque fue más común en los de bajo rendimiento.

Un análisis adicional identificó cuatro patrones utilizados por los empleados de alto rendimiento:

- Constructores de comunidades: este grupo se enfoca en comunidades de práctica, de base y pruebas de concepto (como plantilla y como semilla, como se describió anteriormente). Esto ocurre el 46% de las veces.
- Universidad: este grupo se enfoca en la educación y la capacitación, y la mayoría de sus esfuerzos se destinan a centros de excelencia, comunidades de práctica y centros de capacitación. Este patrón solo se observó el 9% de las veces, lo que sugiere que, si bien esta estrategia puede tener éxito, no es común y requiere una inversión y una planificación significativas para garantizar que las lecciones aprendidas se amplíen en toda la organización.
- Emergente: este grupo se ha centrado en los esfuerzos de base y las comunidades de práctica. Este parece ser el grupo más no intervencionista y aparece en el 23% de los casos.
- Experimentadores: Los experimentadores aparecieron en el 22% de los casos. Este grupo tiene altos niveles de actividad en todas las estrategias excepto big bang y dojos, es decir, todas las actividades que se enfocan en la comunidad y la creación. También incluyen altos niveles en PoC que se estancan. El hecho de que puedan aprovechar esta actividad y seguir teniendo un alto rendimiento sugiere que utilizan esta estrategia para experimentar y probar ideas rápidamente.

Principios de la gestión eficaz del cambio organizacional

Todas las organizaciones son complejas y cada organización tiene diferentes objetivos, un punto de partida diferente y sus propias formas de abordar los desafíos. Las recetas que funcionan en una organización pueden no mostrar los mismos resultados en otra organización. Sin embargo, su organización puede seguir algunos principios generales para aumentar sus posibilidades de éxito.

El trabajo de mejora nunca se hace

Las organizaciones de alto desempeño nunca están satisfechas con su desempeño y siempre están tratando de mejorar en lo que hacen. El trabajo de mejora está en curso y se integra en el trabajo diario de los equipos. Las personas en estas organizaciones entienden que fallar en el cambio es tan riesgoso como el cambio, y no usan "así es como siempre lo hemos hecho" como una justificación para resistirse al cambio. Sin embargo, eso no significa adoptar un enfoque indisciplinado para el cambio. La gestión del cambio debe realizarse de manera científica en busca de un equipo medible o un objetivo organizacional.

Los líderes y los equipos acuerdan y comunican resultados medibles, y los equipos determinan cómo lograrlos

Es esencial que todos en la organización conozcan los resultados empresariales y organizativos medibles por los que están trabajando. Estos resultados deben ser breves (unas pocas oraciones como máximo) a nivel organizacional y coincidir claramente con el propósito y la misión de la organización. A nivel de una unidad de negocio individual, los resultados deben caber en una sola página. Los resultados de la organización deben ser decididos por los líderes y los equipos que trabajan juntos, aunque los líderes tienen la máxima autoridad. En los niveles más bajos de la organización, las metas se establecen con más detalle y con horizontes más cortos.

Sin embargo, debe depender de los equipos decidir cómo van a lograr estos resultados, por las siguientes razones:

- En condiciones de incertidumbre, es imposible decidir el mejor curso de

acción a través de la planificación únicamente. Eso no significa que cierto nivel de planificación no sea importante. Pero los equipos deben estar preparados para modificar o incluso reescribir el plan en función de lo que descubran al intentar ejecutarlo.

- Cuando a las personas se les dice qué hacer y cómo hacerlo, pierden su autonomía y la oportunidad de aprovechar su ingenio. Esto no solo produce peores resultados, sino que también conduce a empleados desmotivados.
- La resolución de problemas es fundamental para ayudar a los empleados a desarrollar nuevas habilidades y capacidades. Las organizaciones deben dar a los equipos problemas para resolver, no tareas para ejecutar.

El cambio a gran escala se logra de forma iterativa e incremental

El ciclo presupuestario anual tiende a impulsar a las organizaciones hacia un modelo basado en proyectos en el que el trabajo de todo tipo está vinculado a proyectos costosos que tardan mucho tiempo en entregarse. Con pocas excepciones, es mejor dividir el trabajo en partes más pequeñas que se puedan entregar de forma incremental. Trabajar en lotes pequeños ofrece una gran cantidad de beneficios. La más importante es que permite a las organizaciones corregir el rumbo en función de lo que descubren. Esto evita perder tiempo y dinero haciendo un trabajo que no brinda los beneficios esperados.

Pasar de un paradigma de proyecto a un paradigma de producto es una tendencia a largo plazo que la mayoría de las industrias tardarán años en ejecutar, pero está claro que este es el futuro. Incluso el gobierno federal de EE. UU. ha experimentado con éxito con la contratación modular para seguir un enfoque incremental más iterativo para entregar grandes piezas de trabajo.

Los problemas que se aplican a la entrega de proyectos también se aplican a la transformación. Las organizaciones deben encontrar formas de lograr ganancias rápidas, compartir el aprendizaje y ayudar a otros equipos a experimentar con estas nuevas ideas.

Errores comunes en la transformación de la cultura

Los líderes a menudo cometen los siguientes errores cuando intentan realizar cambios a gran escala en una organización.

- Tratar la transformación como un proyecto de una sola vez. en las organizaciones de alto desempeño, mejorar es un esfuerzo continuo y parte del trabajo diario de todos. Sin embargo, muchos programas de transformación se tratan como eventos únicos a gran escala en los que se espera que todos cambien rápidamente la forma en que trabajan y luego continúen con el negocio como de costumbre. A los equipos no se les da la capacidad, los recursos o la autoridad para mejorar la forma en que trabajan, y su desempeño se degrada gradualmente a medida que los procesos, las habilidades y las capacidades del equipo se adaptan cada vez menos a la realidad cambiante del trabajo. Debe pensar en la transformación de la tecnología como una parte importante de la entrega de valor del negocio, una en la que no dejará de invertir. Después de todo, ¿planea dejar de invertir en la adquisición de clientes o en la atención al cliente?
- Tratar la transformación como un esfuerzo de arriba hacia abajo. En este modelo, las líneas de informes organizativos se modifican, los equipos se trasladan o se reestructuran y se implementan nuevos procesos. A menudo, a las personas afectadas se les otorga poco control sobre los cambios y no se les brinda la oportunidad de realizar aportes. Esto puede causar estrés y pérdida de productividad a medida que las personas aprenden nuevas formas de trabajar, a menudo mientras aún están cumpliendo con los compromisos existentes. Cuando se combina con la comunicación deficiente que es frecuente en las iniciativas de transformación, el enfoque de arriba hacia abajo puede hacer que los empleados se sientan descontentos y desconectados. También es poco común que el equipo que está planificando y ejecutando la transformación de arriba hacia abajo recopile comentarios sobre los efectos de su trabajo y realice los cambios correspondientes. En cambio, el plan se ejecuta sin importar las consecuencias.

- No ponerse de acuerdo y comunicar el resultado previsto. A veces, las transformaciones se ejecutan con objetivos mal definidos o con objetivos cualitativos (en lugar de cuantitativos), como "tiempo de comercialización más rápido" o "costos más bajos". A veces, los objetivos están definidos pero no son alcanzables, o los objetivos enfrentan a una parte de la organización con la otra. En estos casos, es imposible saber si el trabajo de mejora está teniendo el efecto deseado. Cuando esta falla se combina con un enfoque de arriba hacia abajo, se vuelve difícil experimentar con otros enfoques que podrían ser más rápidos o más baratos. El resultado suele ser un desperdicio cuando se ejecuta el plan y la incapacidad de determinar si se logró la meta o si funcionó el programa. En muchos casos, en lugar de analizar críticamente el fracaso y utilizarlo como una oportunidad de aprendizaje, se ignora el fracaso y se inicia una nueva iniciativa de cambio total o se desacreditan metodologías completas.

La combinación de tratar la transformación como un proyecto y tratarla como una iniciativa de arriba hacia abajo tiende a conducir al patrón que se muestra en el siguiente diagrama. El rendimiento se degrada gradualmente. Al comienzo de un programa de transformación, inicialmente empeora antes de mejorar. Pero luego esto es seguido por una transición de regreso a los negocios como de costumbre. Mientras tanto, el cinismo y la falta de compromiso aumentan en toda la organización.

Catálogo de capacidades

El catálogo de capacidades es un directorio de áreas de conocimiento que define y explora las capacidades técnicas, de proceso, de medición y culturales que impulsan una mejor entrega de software y más alto rendimiento organizacional.

Las capacidades se pueden categorizar en 4 áreas contextos principales:

- **Capacidad técnica**
- **Capacidad de producto y proceso**
- **Capacidad de monitoreo y medición**
- **Capacidad cultural**

Cada uno de los enlaces a continuación presenta una capacidad, analiza cómo implementarla y cómo superar los obstáculos comunes.

El catálogo de capacidades es autoría del equipo de Google Cloud y el proyecto de investigación DORA. En esta guía de adaptación solo listaremos las capacidades, para ampliar en su contenido (recomendado) también se presentará el enlace como adjuto al título.

Capacidad técnica

- Arquitectura
- Automatización de despliegues
- Control de versiones
- Desarrollo basado en troncos
- Desplazamiento a la izquierda en seguridad
- Entrega continua
- Gestión de cambios a bases de datos
- Gestión de data de prueba
- Infraestructura en la nube
- Integración continua
- Mantenibilidad de código
- Pruebas continuas
- Empoderar a los equipos para elegir herramientas

Capacidad de producto y proceso

- Experimentación de equipo
- Agilización de la aprobación de cambios
- Retroalimentación de los clientes
- Visibilidad del trabajo en el flujo de valor
- Trabajando en lotes pequeños

Capacidad de monitoreo y medición

- Monitoreo y observabilidad
- Sistemas de monitoreo para informar decisiones de negocios
- Notificación proactiva de fallas
- Límites de trabajo en proceso
- Capacidades de gestión visual

Capacidad cultural

- Satisfacción laboral
- Liderazgo transformacional
- Cultura de aprendizaje

Casos de estudio

GBH es una compañía de soluciones tecnológicas en República Dominicana, con mayor incidencia en los servicios de desarrollo de software customizado y operaciones de TI. Fundada en el 2005, cuenta con más de 18 años de experiencia en una gran variedad de industrias y áreas de interés. Actualmente posee más de 150 colaboradores y se posiciona como una de las organizaciones de más prestigio a nivel nacional.

Su viaje evolutivo de DevOps empezó en el 2018 al convertirse en la primera organización en República Dominicana en prestar suficiente atención como para introducir entre sus rangos el rol de un ingeniero DevOps; justamente para liderar la transformación cultural, de automatización, y procesos que necesitaba.

Debido a la naturaleza y los objetivos del negocio particulares de GBH, la normalización y estandarización de tecnologías no fue un objetivo fácil de lograr dado que los servicios que ofrece se adaptan a las realidades y necesidades de otros clientes. Sin embargo, se identificaron cuáles eran los "pain points" comunes para tanto el proveedor (GBH) como el cliente.

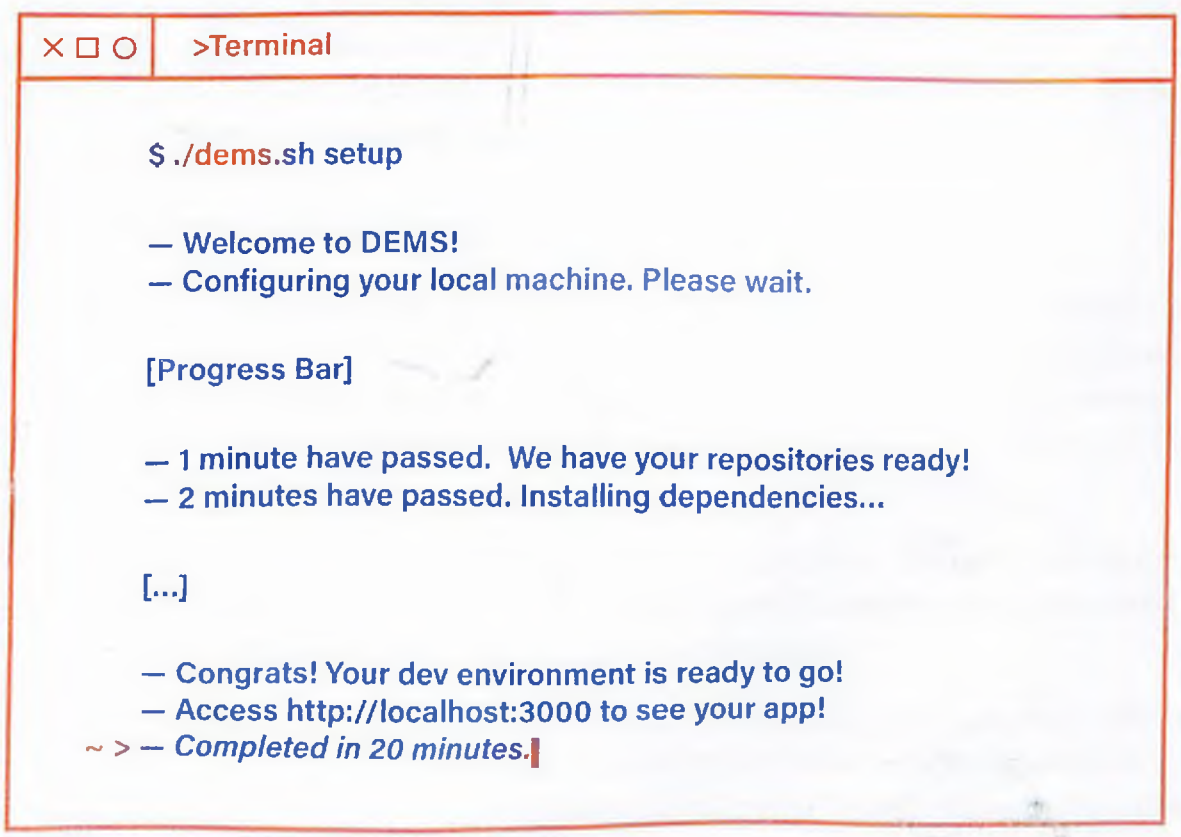
Uno de los primeros esfuerzos que tuvieron sentido fue la creación de un conjunto de scripts a los cuales se les llamó "Dockerized" por tener una dependencia con Docker,

el motor de contenerización más popular actualmente. Casi en paralelo, se trabajó con la primera implementación de pipelines de integración y despliegue. El éxito de estas iniciativas rápidamente captó la atención de todos los equipos y de los líderes de ingeniería. Aquí fue donde empezó la transformación cultural de GBH.

Dockerized, DEMS

Dockerized, y su posterior evolución DEMS (que viene de: Development Environment Management System, Sistema de gestión de ambientes de desarrollo en español), fue una de las primeras demostraciones de éxito de manera aislada que tuvo GBH para uno de sus más importantes equipos en el 2018. Este éxito esparció rápidamente a otras personas y clientes, lo que facilitó la inversión en iniciativas de estandarización y automatización más adelante.

¿Qué exactamente logró Dockerized en sus inicios? Pues, de manera resumida, logró reducir el tiempo de onboarding al proyecto de 4-5 días en promedio, a solo 30 minutos. Estamos hablando de una reducción del 8000% del tiempo que originalmente se requería.



La situación particular que logró intervenir Dockerized, partió de los siguientes "pain points" para el equipo y la organización en general:

- El onboarding técnico al proyecto tomaba una semana de trabajo compartido. Los miembros existentes tenían que instruir al nuevo miembro sobre los requerimientos, dependencias y configuraciones que necesitaría instalar antes de correr la aplicación, y asistir en cualquier dificultad que este encontrase. Todo era manual y no existía ningún tipo de documentación.
- La mayor parte del tiempo de onboarding se despreciaba en resolución de inconvenientes en la configuración de ambiente de desarrollo. Es decir, el desarrollador invertía días de trabajo tan solo logrando hacer funcionar su ambiente local de trabajo en su computador.
- La aplicación solo era configurable a través de un snapshot de una máquina virtual. Dicho snapshot carecía de instrucciones o documentación de uso, por lo que una persona sin previa experiencia en VMs se le haría cuesta arriba garantizar la funcionalidad sin asistencia de algún miembro equipo.

Tomando en cuenta todos estos puntos, se decidió aprovechar los conceptos básicos de scripting y la tecnología de conterización para lograr portabilizar el sistema y que su configuración fuera un asunto tan trivial como la ejecución de un comando. Y así fue: Con tan solo ejecutar un script y un poco tiempo de espera, ya era posible aprovisionar un ambiente de desarrollo totalmente funcional de manera automatizada.

CI para aplicaciones móviles

Tras el indiscutible éxito de Dockerized (DEMS), se logró afianzar aún más la confianza en las iniciativas orientadas en prácticas DevOps. Así fue como surgieron las primeras implementaciones de pipelines de integración continua y despliegues automatizados. Un hito importante alcanzado gracias a CI fue la configuración del primer pipeline para aplicaciones móviles.

Aún en el 2018, la información disponible para los ambientes de desarrollo móvil era bastante escasa. Sin embargo, GBH tenía una deficiencia en sus procesos de desarrollo y aseguramiento calidad para aplicaciones móviles.

En equipos compuestos por más de un desarrollador, y con capacidad para un solo ingeniero de QA, la velocidad de desarrollo no compensaba la velocidad de prueba, sobre todo cuando las compilaciones y simulaciones eran totalmente manuales. Por

día (8 horas), un ingeniero QA solo podía probar máximo 4 cambios; mientras que los ingenieros de desarrollo producían casi el triple de esta cantidad.

El problema era evidente: compilaciones manuales. GBH entonces diseñó un flujo automatizado en el que se logró que la compilación se realizara en demanda por nodos aprovisionados en AWS, y los artefactos requeridos fueran almacenados y expuestos a través de códigos QR, lo que facilitaría el proceso de instalación para los ingenieros QA.

Con esto, se logró una optimización de aproximadamente el 300% en velocidad de prueba. También se obtuvo un beneficio adicional: el equipo completo podía instalar el mismo artefacto y validar cualquier defecto o error encontrado.

Lecciones aprendidas de GBH

El éxito de una práctica DevOps puede generar mayor impacto si es aplicada en equipos pequeños.

Se deben poner atención a los problemas más comunes que afectan la productividad del equipo u organización.

NETFLIX

Netflix es bien conocido como practicante de DevOps. Sin embargo, no considera DevOps a propósito. Este caso de estudio analiza cómo Netflix implementó DevOps basándose en sus principios y enfatizando una cultura colaborativa que valora la innovación.

A pesar de que **Netflix** es una empresa de entretenimiento, no se ha quedado rezagada con respecto a varias de las principales empresas de TI en términos de avances tecnológicos. Netflix ha tenido un gran impacto en el mundo tecnológico a lo largo de los años con su única aplicación de transmisión de video, esfuerzos de ingeniería de clase mundial, cultura y desarrollo de productos.

DevOps es uno de esos métodos que Netflix ejemplifica admirablemente. Su cultura DevOps les ha permitido desarrollarse más rápidamente, lo que se ha traducido en numerosos beneficios comerciales. También los ayudó a lograr un tiempo de disponibilidad casi perfecto, brindar nuevos servicios a los consumidores más rápidamente y aumentar sus suscriptores y horas de transmisión.

Netflix es el servicio de transmisión más popular del mundo en la actualidad, con casi **214 millones de clientes y transmisión en más de 190 países**. Gran parte de su éxito se puede atribuir a su capacidad para adaptar la tecnología más nueva y su cultura DevOps, que le permite desarrollarse rápidamente para satisfacer las necesidades de los consumidores y mejorar las experiencias de los usuarios. Sin embargo, Netflix no cree en DevOps.

Netflix y su migración a la nube

Todo comenzó con la peor interrupción en la historia de Netflix cuando se enfrentó a una importante corrupción de la base de datos en 2008 y no pudo enviar DVD a sus miembros durante tres días. En ese momento, Netflix tenía aproximadamente 8.4 millones de clientes y un tercio de ellos se vieron afectados por la interrupción. Hizo que Netflix se mudara a la nube y le diera a su infraestructura un cambio de imagen completo. Netflix eligió a AWS como su socio en la nube y tardó casi siete años en completar su migración a la nube.

Netflix optó por reescribir toda la aplicación en la nube para volverse verdaderamente nativa de la nube, lo que cambió fundamentalmente la forma en que operaba la empresa.

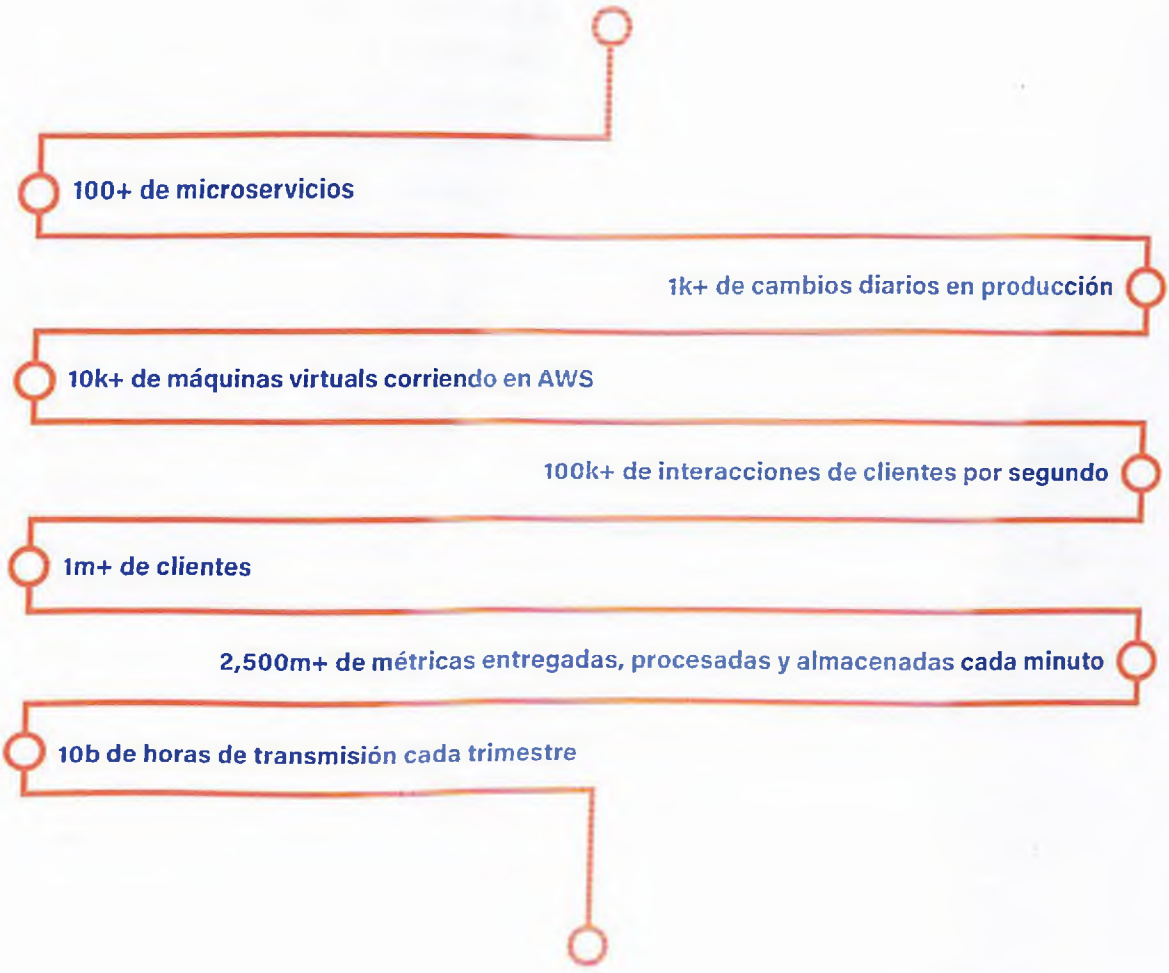
Como parte importante de su transformación, Netflix convirtió su aplicación Java monolítica basada en el centro de datos en una arquitectura de microservicios Java basada en la nube.

Provocó los siguientes cambios:

- **Modelo de datos desnormalizados usando bases de datos NoSQL**
- **Permitió que los equipos de Netflix se acoplaran libremente**
- **Permitió a los equipos crear e impulsar cambios a la velocidad con la que se sentían cómodos.**
- **Coordinación de liberación centralizada**
- **Los ciclos de aprovisionamiento de hardware de varias semanas llevaron a una entrega continua**
- **Los equipos de ingeniería tomaron decisiones independientes utilizando herramientas de autoservicio**

Como resultado, ayudó a Netflix a acelerar la innovación y tropezar con la cultura DevOps. Netflix también ganó ocho veces más suscriptores que en 2008. Y las horas mensuales de transmisión de Netflix también crecieron mil veces desde diciembre de 2007 hasta diciembre de 2015.

Después de completar su migración a la nube a AWS en 2016, Netflix tenía:



Y manejó todo lo anterior con 0 centros de operaciones de redes y unos 70 ingenieros de operaciones.

Datos curiosos: El 21 de abril de 2011, AWS experimentó una gran interrupción en la región Este de EE. UU., pero la transmisión de Netflix funcionó sin interrupciones. Y el 24 de diciembre de 2012, AWS enfrentó problemas en los servicios de Elastic Load Balancer (ELB), pero Netflix no experimentó un apagón inmediato. El sitio web de Netflix estuvo activo durante la interrupción y admitió la mayoría de sus servicios y transmisión, aunque con una latencia más alta en algunos dispositivos.

Lecciones aprendidas de Netflix

- Centrarse en dar libertad y responsabilidad a los ingenieros: Netflix tiene como objetivo contratar personas inteligentes y brindarles la libertad de resolver los problemas de la manera que mejor les parezca. Por lo tanto, no tiene que crear restricciones y barandillas artificiales para predecir lo que deben hacer sus desarrolladores. Pero en su lugar, contrate a personas que puedan desarrollar un equilibrio de libertad y responsabilidad.
- No crear silos, paredes y cercas: Los equipos de Netflix saben dónde encajan en el ecosistema, su trabajo con otros equipos, dependientes y dependencias.
- Siempre ponga la satisfacción del cliente primero: El objetivo final de DevOps es orientarse al cliente y centrarse en mejorar la experiencia del usuario con cada lanzamiento.
- No haga DevOps, pero concéntrese en la cultura: En Netflix, DevOps surgió como el maravilloso resultado de su cultura, pensamiento y prácticas saludables.

Conclusiones

Esta guía de adoptación reconoce que toda organización, tomando en cuenta su cultura e individuos, es totalmente única. Y por tanto, sus problemas, necesidades y objetivos también lo son. DevOps no trata de ser un movimiento milagroso que se adapta a la perfección en la mayoría de los casos, es un proceso evolutivo que será también exclusivo y propio.

Al ver y alinear nuestra visión sobre cómo empieza este proceso de transformación, tomando en cuenta la información recolectada por diferentes fuentes patrocinados por el proyecto de investigación **DORA** en cuánto los factores de éxito de muchas organizaciones en el mundo, podemos afirmar que es totalmente posible alcanzar el éxito con DevOps con un gran retorno de inversión siguiendo las pautas y consideraciones que hemos visto a lo largo de esta guía.

Es importante reconocer que es totalmente válido fallar, tomar decisiones no muy óptimas o no ver el éxito en algunas iniciativas iniciales. No hay por qué empezar con lo perfecto, DevOps en esencia también habla de continuidad en todos sus aspectos.

¡Buena suerte en tu viaje evolutivo! 👊

UNIVERSIDAD NACIONAL PEDRO HENRÍQUEZ UREÑA
FACULTAD DE CIENCIAS Y TECNOLOGÍA
ESCUELA DE INFORMÁTICA

TRABAJO DE GRADO

"Diseño de Guía de Adaptación de Prácticas DevOps para Empresas con Procesos de Ingeniería de Software".

UNPHU

Proyecto de grado presentado por:

Ángel Manuel Adames Montaño

Ing. José Ramón Romero
Miembro del jurado

Ing. Ambiorix Liriano
Miembro del jurado

Ing. Hugo Parada Leal
Miembro del jurado

Ing. Pantaleón Mueses
Asesor

Ing. Héctor Santillán
Director

UNIVERSIDAD NACIONAL PEDRO HENRÍQUEZ UREÑA
ESCUELA DE INFORMÁTICA
Santo Domingo, R.D.

Fecha de sustentación: Agosto 30, 2022.