UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

DevOps, Integridad y Agilidad Continúa

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Unidad 5: Casos

Presentación:

En esta unidad se presentaran casos de éxito en organizaciones que adoptaron una cultura de DevOps, algunas de ella con distintas prácticas de acuerdo a lo que su Negocio lo requiera o en algunos casos dependiendo de la tecnología que utilizan. Además se identificaran distintas prácticas que utilizadas por ellos.

Objetivos:

Que los participantes:

- Conozcan los principales casos de éxito de organizaciones que utilizan DevOps.
- Conozcan los circunstancias de la utilización de ciertas prácticas en ciertas organizaciones
- Identifiquen la importancia de acompañar las incorporación de prácticas con decisiones estratégicas.

Bloques temáticos:

1. **Netflix**
2. **Facebook**
3. **Amazon**
4. **Localytics**
5. **Google**

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

Tomen nota

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Netflix

Netflix es un caso fantástico en la manera en que pasaron de ser una compañía que rentaba DVD a transformarse en una empresa para un producto digital (streaming) de alta calidad como lo conocemos actualmente. Para el modelo de empresa actual, se basaron en un desarrollo de software Ágil en conjunto a una cultura DevOps para toda la organización. El enfoque DevOps que adoptó Netflix fue apoyarse en la automatización tanto para su infraestructura como para las pruebas. El producto está montado sobre la plataforma de Servicios Web de Amazon (AWS), la decisión de llevar la infraestructura a la nube radica en una decisión de negocio. Los ingenieros de Netflix analizaron que la gran cantidad de dispositivos en los que los usuarios podría utilizar para el consumo del streaming era de tal magnitud que ante cambios a nivel servidor y cliente (navegadores web, smartphones, consolas de vídeo juegos, etc.) el impacto que podría tener, dificultaría asegurar la calidad y la experiencia de usuario. Para ello, se apoyaron en la confiabilidad de infraestructura versionada, arquitectura de microservicios, integración continua, entrega continua y automatización de pruebas en todos los ámbitos: desarrollo, configuración, infraestructura.

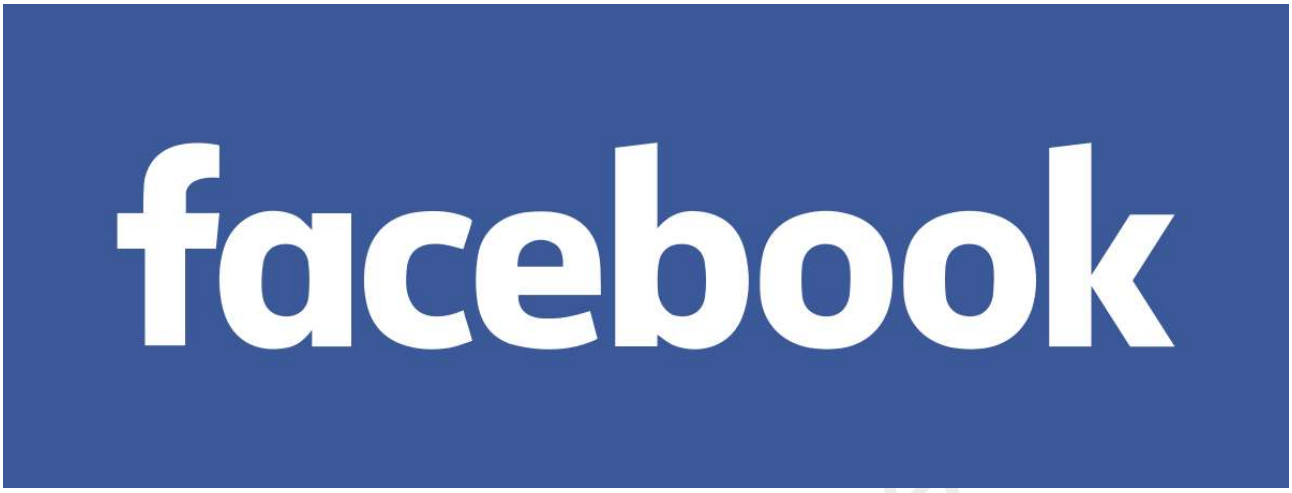
La visión de DevOps de está compañía, dadas sus necesidades de negocio, está enfocada principalmente en la entrega de valor a sus clientes, en la calidad del software que construyen y en la mejora continua. El enfoque novedoso que consiguió Netflix fue hacer una cultura de la ***automatización de las fallas***, de manera que el producto tenga una alta robustez y fiabilidad en los servidores y clientes. De esta manera lo que construyeron fueron scripts de automatización para que ciertos servicios fallen de una manera aleatoria, en los entornos de desarrollo, generando escenarios de prueba en los que los desarrolladores tienen en consideración de manera que dichos fallos no afecten el servicio con el usuario. Por ejemplo, en la aplicación de Netflix hay una sección que es la de “recomendados”, si por alguna razón el servicio se cae esto no afecta la experiencia de usuario ni lanza un error, sino que en ciertas ocasiones aparece otra funcionalidad similar que la reemplaza temporalmente o directamente no aparece nada, pero la aplicación nunca deja de funcionar. Lo mismo, por ejemplo, en el momento de la reproducción. Si hay fallas en algún punto de la aplicación, el streaming está desarrollado de tal manera que adapta la calidad para intentar que el usuario no pierda segundos en la reproducción. Por lo tanto, los desarrolladores no sólo construyen funcionalidades en la aplicación sino que también construyen un sistema altamente tolerante a las fallas.

El sistema que introdujeron los ingenieros de Netflix para la automatización de diferentes tipos de fallas se llama “Ejército de Simios” (Netflix Simian Army) provocó un cambio en la

proceso del diseño, desarrollo, pruebas e implementación de software en la compañía ya que los componentes del software son generados para tener un alto grado de modularidad y resiliencia a fallas, dando como resultado software de muy alta calidad y confianza antes inconvenientes en el “back-end” de la aplicación. Pero no fue la herramienta la que cambió la concepción de la manera de desarrollar sino que fue la manera en que la organización “piensa” su producto y su negocio. La compañía, se centra en el cliente (Customer Centric), busca distintas maneras de mejorar y adaptarse a las necesidades que poseen sus clientes con respecto a sus productos y a sus objetivos de negocio. De este tipo de pensamiento es que surgen las ideas de cómo darle una buena experiencia al usuario, de no tener cortes en el servicio y en caso de tenerlos minimizar su impacto.

Netflix utiliza una gran variedad de lenguajes de programación y frameworks para dichos lenguajes, así como también una variedad de prácticas y herramientas para construir y desplegar sus aplicaciones en los servidores de Amazon. Algunas tecnologías que utiliza son Java, Jenkins, Gradle (builds para Java), Infraestructura Inmutables y Spinnaker para sus deploys en AWS (todo cambio de ambiente se realiza a través de un pipeline automatizado). Toda esta diversidad de conceptos del mundo de la tecnología surgieron al concientizar que el modelo de negocio al que querían ir debía adoptar un cambio organizacional mediante una cultura DevOps; en la que la medición para la inspección y adaptación de su proceso y negocio eran claves para llevar a cabo su estrategia comercial a través de un producto digital. Netflix adoptó la arquitectura de microservicios en tal grado de madurez que organizacionalmente refleja parte de sus equipos de desarrollo en cada o conjunto de microservicios, es decir que divide sus equipos en pequeñas células que reflejen el funcionamiento del servicio.

2. Facebook



En Facebook, lugar de planificar el trabajo en proyectos o dividir el trabajo en Sprints en tiempo real, los desarrolladores realizan la mayor parte de su trabajo en pequeños cambios independientes que se lanzan con frecuencia. Esto tiene sentido en el modelo de negocio de la compañía, todos están sincronizando constantemente y probando nuevas

opciones y aplicaciones en diferentes comunidades de usuarios, viendo qué se implementará. Es un crédito para su arquitectura que tantos pequeños e independientes cambios desarrollados por los programadores puedan hacerse de forma independiente y económica.

Con el tiempo, la industria del software ha presentado varias formas de entregar el código de manera más rápida, segura y con mejor calidad. Muchos de estos esfuerzos se centran en ideas tales como la Integración Continua, la Entrega Continua, el desarrollo ágil, DevOps y el desarrollo basado en pruebas (BDD). Todas estas metodologías tienen un objetivo común: permitir que los desarrolladores obtengan feedback de manera rápida y correcta de las personas que lo usan, en pasos seguros, pequeños e incrementales. Los procesos de desarrollo y despliegue en Facebook han crecido orgánicamente para abarcar muchas partes de estas técnicas de iteración rápida sin adherirse rígidamente a ninguna en particular. Este enfoque flexible y pragmático ha permitido lanzar sus productos web y móviles exitosamente en diferentes calendarios y husos horarios.

Durante muchos años, las implementaciones del llamado “front-end” de Facebook se realizaban tres veces al día usando la estrategia de branching de utilizar una rama principal (master/trunk) y desplegando a producción a través de release branch. Los ingenieros seleccionaban los cambios deseados mediante la técnica de “cherry-picks” (cambios en el código que habían pasado una serie de pruebas automáticas) para incorporarlos de la rama master en uno de las integraciones diarias de la rama de release. En general, Facebook poseía entre 500 y 700 selecciones de cambios por día y una vez a la semana, generaban un nuevo branch release que recogía los cambios que no fueron seleccionados durante la semana. Los desarrolladores también son responsables de escribir las Pruebas Unitarias (UnitTest), sus propias pruebas de regresión y pruebas de rendimiento automatizadas. Los desarrolladores también prueban el software usando la versión de desarrollo de la aplicación para su uso personal. También, Facebook depende de los clientes para probar el software. El software se publica en tandas para las pruebas A / B y "experimentación en vivo" en subconjuntos de la base de usuarios, ya sea que los clientes quieran participar o no en esta prueba (utilizan la técnica de Release Canary). Debido a que su base de clientes es tan grande, puede obtener comentarios significativos de las pruebas con incluso un pequeño porcentaje de usuarios, lo que al menos minimiza el riesgo y un impacto negativo para los clientes.

En 2016, Facebook vio que la estrategia de release branch y cherry-picks estaba llegando a su límite. Se evaluaban más de 1000 diferencias de código por día en la rama

master, y la integración semanal era a veces de hasta 10000 diferencias. La cantidad de esfuerzo manual necesaria para coordinar y entregar una publicación era tan grande cada semana que no fue sostenible.

El sitio facebook.com decidió cambiar a un sistema "push from master" de entrega casi continua en abril de 2016. Durante el año siguiente, su implementación fue gradualmente, primero al 50 por ciento de los empleados, luego del 0.1% al 1% y al 10% del tráfico web de producción. Cada una de estas progresiones permitió probar la capacidad de la organización, infraestructura y procesos para manejar una mayor frecuencia de implementaciones a producción y obtener feedback del mundo real. El principal objetivo era asegurar que el nuevo sistema mejorará la experiencia del usuario, o al menos no empeorarla. Después de casi exactamente un año de planificación y desarrollo, en el transcurso de abril de 2017, Facebook habilitó el 100% de los servidores web de producción para ejecutar el código implementado directamente desde la rama master.

Si bien un verdadero sistema de Continuous Deployment entregaría cada cambio individual a producción poco después de desarrollarlo, la velocidad del código en Facebook obligó a desarrollar un sistema que empuje de decenas a cientos de diferencias de incrementos de producto desarrolladas cada pocas horas. Los cambios que se realizan en este modo de entrega cuasi continuo son generalmente pequeños e incrementales, y muy pocos tendrán un efecto visible en la experiencia real del usuario. Cada release se implementa al 100% en producción de forma escalonada en unas pocas horas, por lo que se puede detener el release si se encontrara algún inconveniente, ya que implementan la técnica de Canary Release.

3. Amazon

Amazon es una de las compañías tecnológicas más rentables de la actualidad. Se transformó en 2006 de un minorista online a un gigante tecnológico y pionero en el espacio en la nube con el lanzamiento de Amazon Web Services (AWS) siendo una oferta de Infraestructura como Servicio (IaaS) bajo demanda. Amazon aceptó muchos riesgos con

este lanzamiento. Al desarrollar uno de los primeros servicios masivos Cloud públicos, aceptaron que muchos de los desafíos serían desconocidos, y muchas de las soluciones no estaban probadas. Para entender y aprender del éxito de está compañía hay que hacerse las preguntas adecuadas ¿Qué pasos tomó Amazon para minimizar los riesgos asociados a estos cambios? ¿Cómo definieron los ingenieros de Amazon su proceso para garantizar la calidad?

Para ello, la visión del CEO de Amazon, Jeff Bezos, tenía sobre la compañía fueron los principios subyacentes de lo que ahora llamamos DevOps, así como su dedicación a lo que se podría decir que son los principales atributos de calidad de la plataforma AWS: interoperabilidad, disponibilidad, confiabilidad y seguridad. Según Yegge (ex ingeniero de Amazon), Jeff Bezos emitió un mandato durante el desarrollo temprano de la plataforma AWS, que declaró, en palabras de Yegge (esto se ha tomado de un documento que accidentalmente Steve Yegge lo hizo público):

- Todos los equipos expondrán sus datos y funcionalidad a través de interfaces de servicio (APIs)
- Los equipos deben comunicarse entre sí a través de estas interfaces.
- No habrá otra forma de comunicación entre procesos permitida: no hay enlaces directos, no hay lecturas directas a la base de datos de otro equipo, no hay modelo de memoria compartida, no hay puertas de atrás de ninguna clase. La única comunicación permitida es a través de llamadas de interfaz de servicio a través de la red.
- No importa qué tecnología se utilice. HTTP, Corba, Pubsub, protocolos personalizados, etc.
- Todas las interfaces de servicio, sin excepción, deben diseñarse desde cero para ser externalizables. Es decir, el equipo debe planificar y diseñar para poder exponer la interfaz a los desarrolladores del mundo exterior. Sin excepciones.
- Cualquiera que no haga esto será despedido.

Prestemos atención a lo que dice entre líneas el documento filtrado. Establece que los procesos dentro de la organización cambiarían; es decir, los ingenieros de Amazon ahora deben desarrollar API de servicios web para compartir todos los datos internamente en toda la organización. Este cambio está diseñado específicamente para incentivar a los equipos a construir software con un alto nivel de calidad, ya que se requiere que estos construyan API útiles, robustas, seguras o podrían recibir reclamos de otros equipos o bien de los directivos, al no poseer la calidad deseada. La disponibilidad y la confiabilidad se aplicarán

de la misma manera. Hay que tener en cuenta que este mandato fue para **toda** la compañía, no solo para los equipos de desarrollo. Por ejemplo, si el área de marketing o contable desea recopilar datos estadísticos para obtener reportes entonces dichas áreas deben adquirir personal para desarrollar una API que consuma los datos del resto de la organización y a su vez exponer los propios datos dentro de la ella.

DevOps nos enseña a evolucionar una organización que fortalece la calidad, seguridad y minimizar riesgos y tiempos muertos de las entregas a producción a través de prácticas como Integración, Entrega e Implementación Continua (en 2013 la compañía hacía una implementación cada 11,6 segudos), pruebas y pipeline automatizados, uso del framework Scrum y eXtreme Programming (XP), Arquitectura de Microservicios, Release Canary. Puede parecer que este escenario posea una versión autoritaria del pensamiento de DevOps. Sin embargo, tiene que ver con la visión de la empresa de calidad y puesta de valor al cliente, aplicando un requisito riguroso de construir con la misma calidad todo el software interno de la compañía tal como sería para exponerlo a un cliente, y esto extendido a todos los equipos dentro de Amazon.

Estas mejoras de la API ocurrieron orgánicamente en Amazon, sin la necesidad de emitir requisitos rigurosos teniendo en cuenta un lineamiento con los equipos que los desarrollarían, ya que estos fueron incentivados a mejorar continuamente sus API para hacer su propio trabajo más efectivo y eficiente. Cuando se lanzó AWS unos años después, muchas de estas mismas API formaban parte de la interfaz pública de la plataforma AWS, que era notablemente completa y estable en el lanzamiento. Este nivel de calidad en el lanzamiento cumplió directamente con los objetivos comerciales al contribuir a las tasas de adopción temprana y al aumento constante de la popularidad de AWS, una plataforma que brindaba a los usuarios un conjunto completo de capacidades potentes, comodidad y confianza inmediatas en un servicio estable y maduro. A su vez los niveles de madurez que poseía la organización le permitieron tener rápidos despliegues a producción para el propio sitio de ventas de Amazon.com, acortando drásticamente los tiempo de despliegues tantos internos como al usuario (cycle time y time to market).

4. Localytics

Localytics es una compañía de análisis y desarrollo de aplicaciones móviles y web. Marcas principales como ESPN, eBay, Fox, Salesforce, RueLaLa y New York Times utilizan sus herramientas de marketing y análisis para entender el desempeño de sus aplicaciones e involucrar a los clientes nuevos y existentes. El software de la compañía, con sede en

Boston, se utiliza en más de 37000 aplicaciones en más de 2,7 billones de dispositivos de todo el mundo (contratada por más de 6000 empresas). Así mismo está empresa soporta su infraestructura en los servidores Amazon (AWS).

La compañía fue por distintos niveles de madurez en el ciclo de desarrollo y pipeline deployment. Al comienzo y para equipos pequeños no requería tanto nivel de gestión de sus ciclos de desarrollo pero al comenzar a crecer comercialmente la compañía se evidenció una dificultad en el escalamiento de sus equipos de desarrollo y en la organización. Para el 2012 tenían tres ingenieros desarrolladores full stack, un sólo servidor de producción alocado en Amazon y un servidor Jenkins para ejecutar pruebas y realizar la integración del nuevo código con el branch “master”. Requiriendo gran cantidad de horas para medir y evaluar las fallas en los ambientes de desarrollo y producción.

Lo que impulsó el primer cambio en el nivel de madurez de Localytics en el camino a la Entrega Continua fue la implementación de la práctica de Blue Green Deployment. La incorporación de está práctica posibilitó la reducción del riesgo en los despliegues a producción dando más certidumbre que no habría diferencias entre un ambiente de desarrollo y el productivo, lo que fue una de las principales razones por las cuales adoptaron esta práctica. Seguido a incorporar una práctica distinta de implementación, Localytics incorporó la práctica de automatizar las pruebas en cada push (comando de git para enviar los cambios a la rama en que se está trabajando) en lugar de hacerlo solamente al hacer merge sobre el branch principal. Esto les permitió obtener feedback al momento que estaban desarrollando en vez de tener que esperar hasta el momento anterior de un despliegue en producción. En conjunción con esto, también se propuso subir el índice de cobertura de código. Es decir que lo que se propusieron fue formalizar la práctica de integración continua, poniendo foco en las pruebas y mejorando su propio proceso para aumentar la calidad del producto que se desarrolla.

En sintonía con el foco a reducir los riesgos y los tiempos de entrega, junto con aumentar la calidad del producto, la formalización de la práctica de Integración Continua determinó que también fue necesario poner foco en la automatización del pipeline. A medida que la empresa fue escalando y se incorporaron más desarrolladores y aumentó la necesidad de implementar y probar más cantidad de requerimientos de clientes fue necesario la automatización de las tareas manuales y repetitivas para la realización de implementación a distintos ambientes (desarrollo, pruebas, pre-producción, producción). Las tareas manuales son propensas a errores y requieren la atención del 100% de la persona asignada, además que al incorporar nuevos desarrolladores estos no conocían cómo hacer

dichas tareas por lo cual había una notable sobrecarga de trabajo y por ende más errores. La automatización de las tareas de deploy (pipeline) junto con una mejora del aprovisionamiento de ambientes redujo considerablemente la cantidad de errores y los tiempos de implementación, con lo que le permitió a los equipos enfocar su tiempo en el desarrollo de nuevas funcionalidades en lugar de hacer otras tareas. Para ello incorporaron el concepto de Pull Request, una vez que solicitaba un pedido de cambio, se hacía una revisión de código (Code Review) y una vez aprobado se implementa el cambio automáticamente en el ambiente que corresponda mediante un click (one click deploy). Esto permitió que los Project Management (PM) no pierdan tiempo en rastrear en qué ambiente estaba disponibilizado un determinado requisito o bien que no ocurran errores al disponibilizar requisitos de manera no coordinada (casos en los que se implementa un requisito en un ambiente y en el mismo ambiente se pisan las implementaciones con requisitos de otros equipos).

La manera en que está compañía incorporó incrementalmente los valores y prácticas del movimiento DevOps fue muy adecuada, dado que dichos ámbitos de mejora fueron desarrollados de manera incremental y orgánicamente, para poder escalar de una manera segura. Fueron atacando obstáculo a obstáculo y organizándose alrededor de la visión que tenían

5. Google

En esta sección vamos a detenernos sólo en algunos aspectos de las prácticas de ingeniería, ya que analizar todos los cambios sería de mucha extensión para el curso y no es su propósito. La intención es mostrar ciertos aspectos que incorporó el gigante Google en sus prácticas y procesos de ingeniería, que tienen foco en la calidad, automatización, colaboración y acercamiento de las necesidades del usuario.

Dentro de está compañía el foco de atención se centró en cómo hacer una experiencia de usuario excelente al, por ejemplo, realizar una búsqueda en el sitio, leer un correo o que otras empresas utilicen sus sistemas de infraestructura en la nube. Para ello, crearon una cultura de colaboración y co-desarrollo transversal a los equipos en la organización. Permitiendo compartir no sólo experiencias en el producto que desarrollaban sino también código fuente de software, prácticas y técnicas que podría ser potencialmente re utilizado para otros productos/equipos.

Como la mayoría de las empresas orientadas a la cultura DevOps y Agilidad, los equipos de desarrollo están orientados a las necesidades del usuario con una fuerte mirada en las pruebas, la mejora continua y en la calidad. Para el sitio de Google.com escalar significó orientar sus procesos y equipos en otra dirección. Por ejemplo, hacia el 2005 hacer una implementación en el sitio principal no era tan sencillo como se podría llegar a imaginar, el equipo más impactado era el que tenía responsabilidad sobre Servidor Web de Google (GWS, por sus siglas en inglés: Google Web Server), ya que se trataba de una aplicación que canalizaba todas las peticiones de Google.com y de otros sitios. Uno de los principales inconvenientes con los que se enfrentaba el equipo de GWS es su asignación al proyecto, no era un equipo 100% enfocado en sus tareas sino que había veces que se encontraba asignado trabajando con otros equipos que desarrollaban distintas funcionalidades de búsqueda de manera independiente. Los inconvenientes principales eran que las integraciones de nuevo código que realizan al sitio principal de Google no eran probadas en su totalidad antes de la implementación en producción, los builds y las pruebas demoraban mucho tiempo y en ocasiones las integraciones entre distintos equipos traía conflictos. Estos inconvenientes no sólo provocaban que ciertas búsquedas en el sitio se demoren más de lo normal o que se encuentren errores provocando no sólo una pérdida de ganancias sino también de confianza con el usuario. El principal síntoma de esta situación es que los desarrolladores no querían pasar código a producción por miedo a que ocurra algún error, ya que al no comprender el sistema en su totalidad no querían introducir cambios.

La determinación que tuvieron los líderes de Google y del equipo de GWS fue la incorporación de la automatización de pruebas, mediante un acuerdo y una regla que determinaron que ningún cambio se introduciría sin que sea acompañado de las respectivas pruebas automáticas. El cambio introducido en los equipos fue de realizar Integración Continua, hacer seguimiento del índice de cobertura de código (la relación entre la cantidad de código escrito y la cantidad de código que posee pruebas). Los resultados que se

obtuvieron fueron excepcionales, se generó uno de los equipos de más alto rendimiento de la compañía integrando código de diversos equipos cada semana, mientras que los desarrolladores introducían complejos cambios al sistema con confianza gracias a la cobertura y salud del código, y en medir y mejorar la calidad del sistema.

Dentro de la creación de una cultura de la colaboración y el co-desarrollo que promueve Google, el equipo GWS extendió sus mejores prácticas a lo largo de la organización promoviendo una cultura de la automatización de pruebas en gran parte de la compañía. Gracias a las iniciativas de este equipo, actualmente los desarrolladores de Google al enviar un cambio a su repositorio de código se ejecutan cientos de pruebas a través de un pipeline automatizado, en el que si dicho cambio pasa todas las pruebas recién se incorporará a la rama principal para luego ser implementado en producción. A su vez, incorporaron a su manera de trabajar el concepto de tener siempre la “línea en verde” refiriéndose a que cualquier inconveniente en las integraciones que haga que los desarrolladores no puedan subir nuevos cambios debe ser lo más prioritario a atender y resolver. Dado está filosofía Lean, también se extendió a los equipos de infraestructura de Google trabajando en cooperación con los desarrolladores, con lo cual cualquier inconveniente que provoque caídas a nivel general debe tener un procedimiento para volver hacía atrás el cambio que provocó el error (roll back) dejando al sistema en un estado saludable.

Luego de 8 años, para el 2013, las prácticas de testing automático e Integración Continua se expandieron en todos los equipos de Google (4000 proyectos con 15000 desarrolladores), en los que todos ellos desarrollan, integran, prueban e implementan su código en producción juntos. Todo el código se encuentra en un único repositorio compartido que contiene millones de archivos los cuales son construidos e integrados continuamente, con una tasa de cambio en el código fuente del 50% cada mes, 40.000 líneas de código son enviadas por día, 50.000 builds por día, 120.000 baterías de testing automático y 75 millones casos de pruebas ejecutados por día.

Bibliografía utilizada y sugerida

- Gene Kim, Jez Humble, Patrick Debois, and John Willis. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. 1era edición. EE:UU: IT Revolution Press, LLC; 2016

Lo que vimos:

En la unidad recientemente vista se exhibieron distintos casos de organizaciones que adoptaron exitosamente una cultura DevOps, en los que el cambio de enfoque a una perspectiva de centrarse en las necesidades del cliente, promover una cultura colaborativa, de transparencia, excelencia técnica y mejora continua produjo grandes resultados tanto de ganancias, clima laboral e imagen con los clientes.

En cada organización se puede apreciar que la incorporación de distintas herramientas y prácticas es la consecuencia del cambio de perspectiva que tiene la organización, ya sea por demorar en responder a las necesidades del usuario o no perder su confianza, o bien a una decisión de que el enfoque organizacional tradicional no promueve del todo el desarrollo personal y profesional de sus integrantes, ni tampoco la promueve la construcción de productos excelentes.