Darío Martínez Bajo
Diego García Quirós
Sonia Fernández Coto

4 Key Metrics

Introducción

Tras seis años de investigación el equipo de DORA ha identificado cuatro métricas importantes clave que indican el nivel de rendimiento que está llevando el equipo de desarrollo en un proyecto, están métricas son:

1. **Plazo medio de entrega para los cambios:** Define la cantidad de tiempo que toma un cambio en entrar a producción.
2. **Frecuencia de implementación:** Cuantifica con qué frecuencia se lanza el producto a producción con éxito.
3. **Tiempo medio de restauración:** Cuánto tiempo le lleva a la organización recuperarse de un fallo en producción.
4. **Cambiar tasa de error**: Indica el porcentaje de los despliegues que causan un fallo en producción.

Las dos primeras métricas miden la velocidad del equipo de desarrollo, mientras que las dos siguientes son una medida de la calidad y estabilidad de un proyecto de desarrollo. Estas 4 métricas pueden ser extraídas de las herramientas que se estén utilizando para manejar el proceso de desarrollo y el control de versiones: Jira, Git, Jenkins…

Estas 4 métricas propuestas por DORA están diseñadas para permitir al equipo de desarrollo alinear su trabajo con los objetivos de negocio del cliente, y además son un estándar para que el cliente pueda tener una visión de alto nivel de cómo sus desarrolladores están trabajando.

Al disponer de las Four Key Metrics, los equipos de desarrollo son conscientes de que están haciendo lo correcto para llevar a cabo los proyectos y para mejorar su negocio.

Plazo medio de entrega para los cambios

Es el tiempo medio que se tarda desde que se entrega código hasta que se despliega en producción.
Algunas organizaciones miden este valor desde que se realiza el commit hasta el merge (fusión) del código en la rama principal que será desplegada en producción.

Sin embargo, podemos distinguir dos contextos principalmente, el primero sería el del desarrollo del producto, en el que el equipo se centra en satisfacer a múltiples clientes y con poca preparación previa, no sabemos el tiempo que nos va a llevar y tiene una gran variabilidad.

Por otra parte, está la entrega del producto, es decir, el momento de implementar, testear y entregar el producto, que es más fácil de medir para el equipo y tiene una variabilidad menor. Esta tabla nos ejemplifica mejor las diferencias entre estos dos contextos:

Darío Martínez Bajo
Diego García Quirós
Sonia Fernández Coto

Product Design and Development	**Product Delivery (Build, Testing, Deployment)**
Create new products and services that solve customer problems using hypothesis-driven delivery, modern UX, design thinking.	Enable fast flow from development to production and reliable releases by standardizing work, and reducing variability and batch sizes.
Feature design and implementation may require work that has never been performed before.	Integration, test, and deployment must be performed continuously as quickly as possible.
Estimates are highly uncertain.	Cycle times should be well-known and predictable.
Outcomes are highly variable.	Outcomes should have low variability.

En resumen, los plazos de entrega de productos más cortos son mejores, ya que permiten una retroalimentación más rápida sobre lo que estamos construyendo y nos permite cambiar de dirección más rápidamente. Los plazos de entrega cortos también son importantes cuando existe un defecto y necesitamos entregar una solución de forma rápida y eficaz.

¿Cómo se calcula?

Cada desarrollo o proyecto se mide de principio a fin y se calcula un promedio de esos tiempos.

Concretamente, SENTRIO presenta en una gráfica que nos permite ver la medición de dicho tiempo de cada una de las tareas y su correspondiente línea de tendencia. Esto ayuda al equipo a poner el foco en las anomalías, siendo éstas las tareas que tienen un tiempo demasiado elevado y, por tanto, han supuesto un mayor tiempo.

¿Como se mejora?

Esta métrica puede ser mejorada por:

- Automatizar los procesos de despliegue (Reducen tiempos de entrega)
- Asegurarse de que los procesos en la plataforma de CI/CD(integración continua y entrega continua) son lo más eficientes posibles.
- Dividir los desarrollos en procesos más pequeños y manejables. Si el equipo tarda demasiado en entregar una parte del proyecto a los clientes, puede que sus necesidades hayan cambiado y el esfuerzo realizado ya no sea útil. Por ello, resulta más beneficioso realizar entregas pequeñas y frecuentes. Además, trabajar con cambios más pequeños facilita que los desarrolladores puedan recibir feedback más rápido e identificar y resolver defectos antes.

- Crear procesos de “Code Review” eficientes para reducir el tiempo de entrega.

Frecuencia de implementación (implementation frequency)

Cuando se trabaja en el desarrollo, interesa que el tamaño del lote sea pequeño, ya que en caso de que algo sea mejorable, o esté mal, se puede mejorar rápidamente para el siguiente lote. No obstante, medir el tamaño de un lote de software no es sencillo. Por lo que utilizamos como cuantificador la **Frecuencia de implementación**. La forma de medir dicha frecuencia es teniendo en cuenta el número de días laborables por semana que se ha logrado una implementación exitosa. Este número debe ser igual o superior a 3. Lo que obliga al desarrollador a invertir tiempo en el proyecto prácticamente todos los días laborables.

Se entiende como implementación exitosa, por defecto, cualquier avance significativo (es decir, que aporte algo al software) y que, además, no genere errores.

Un error común, es medir la frecuencia de implementación como el número medio de aportaciones. Esto es un error. Esa métrica se corresponde con el volumen de implementación. No con la frecuencia.

Las principales ventajas que nos aporta el tener en cuenta esta métrica, son las siguientes:

1. **Reducción de ciclos de trabajo y riesgo:** Los ciclos cortos son especialmente beneficiosos ya que permiten obtener una respuesta rápida sobre el producto desarrollado y reparar errores o realizar mejoras en el siguiente ciclo. De lo contrario esas posibles mejoras o bugs serían arrastrados en el tiempo.
2. **Mejora de la eficiencia:** De la mano con el punto anterior, las mejoras rápidas y continuas construyen un sistema más eficiente
3. **Mejora la motivación y la urgencia:** Marcando ciclos cortos, los desarrolladores generan objetivos a corto plazo. Lo que incrementa su concentración y compromiso con el proyecto. (No es lo mismo deber tener algo listo para dentro de un par de semanas que para dentro de varios meses)
4. **Reduce los costes.**

Tiempo medio de restauración (Mean time to restore o MTTR)

Esta métrica se refiere al tiempo que suele llevar reparar el sistema sobre el que se trabaja de fallos. Tenerla en cuenta nos aporta un sistema más robusto y resiliente. Esto quiere decir que, ante posibles errores, el sistema puede recuperarse de forma rápida y eficiente. Es especialmente interesante ya que, en lo que a software se refiere, los problemas son prácticamente inevitables y hay que estar siempre preparado para hacerles frente. Para medir este tiempo, es necesario tener en cuenta 3 factores:

1. Cuando se creó el incidente.
2. Cuando se resolvió.
3. Cuando una implementación resolvió dicho incidente.

De esta forma, tendremos varios tiempos de restauración y calculando su media nos hacemos con el resultado de la métrica, que, obviamente, nos interesa que sea bajo.

Algunos consejos para mejorar este dato son:

- Monitorizar el sistema de forma adecuada para saber cuándo exactamente se genera un fallo
- Priorizando la recuperación del sistema ante cualquier tarea para que no se generen fallos mayores y se acelere el proceso de recuperación.

Cambiar tasa de error (change fail rate)

La tasa de error se corresponde al porcentaje de cambios en la producción que generan fallos. Aportaciones fallidas. Es especialmente importante mantener esta tasa lo mas baja posible ya que genera enormes pérdidas de tiempo. Malgastamos tiempo en reparar errores y fallos que podríamos invertir en mejorar o incrementar la funcionalidad del sistema generando deuda técnica.

Esta tasa se calcula como el número de implementaciones que provocan fallo entre el numero de implementaciones totales.

Es especialmente interesante tener una tabla de implementaciones y una de fallos que se unen con el id de la implementación fallida para tener una mejor visión del proceso y de qué arreglar y por qué.

Mejorar esta tasa, se puede lograr mediante las siguientes prácticas:

- Asegurar que todo el código está cubierto por test antes de dar por terminada la implementación
- Realizar Code Reviews ya que un revisor puede ver fallos que el desarrollador a simple vista no ve
- Realizar análisis y revisión de dependencias para evitar fallos de incompatibilidad y versionado.

El pipeline de las cuatro llaves

El canal de ETL encargado de extraer, transformar y cargar los datos para este tipo de métricas.

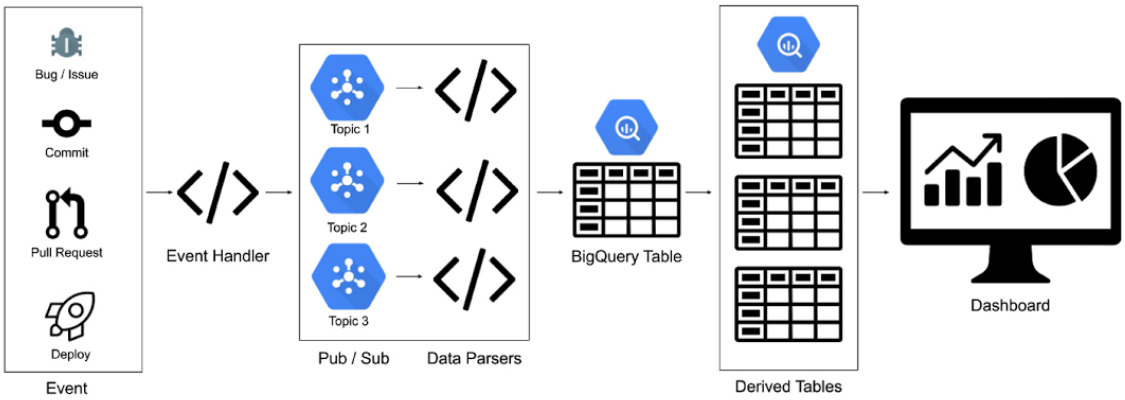
El mayor desafío para poder recopilar este tipo de métricas es que para cualquier equipo los datos de implementación, cambios e incidentes suelen estar en diferentes sistemas muy distintos entre ellos. ¿Cómo desarrollamos una herramienta de código abierto que pueda capturar datos de estas diferentes fuentes, así como de fuentes que quizás desee utilizar en el futuro?

Con Four Keys, nuestra solución fue crear una canalización generalizada que se puede ampliar para procesar entradas de una amplia variedad de fuentes. Cualquier herramienta o sistema que pueda generar una solicitud HTTP se puede integrar en la canalización de Four Keys.

1. Los eventos se envían a un webhook alojado en Cloud Run. Los eventos son cualquier cambio que ocurra en su entorno de desarrollo (por

ejemplo, GitHub o GitLab) que se puede medir, como una solicitud de incorporación de cambios o un problema nuevo. Four Keys define eventos para medir, y puede agregar otros que sean relevantes para su proyecto.

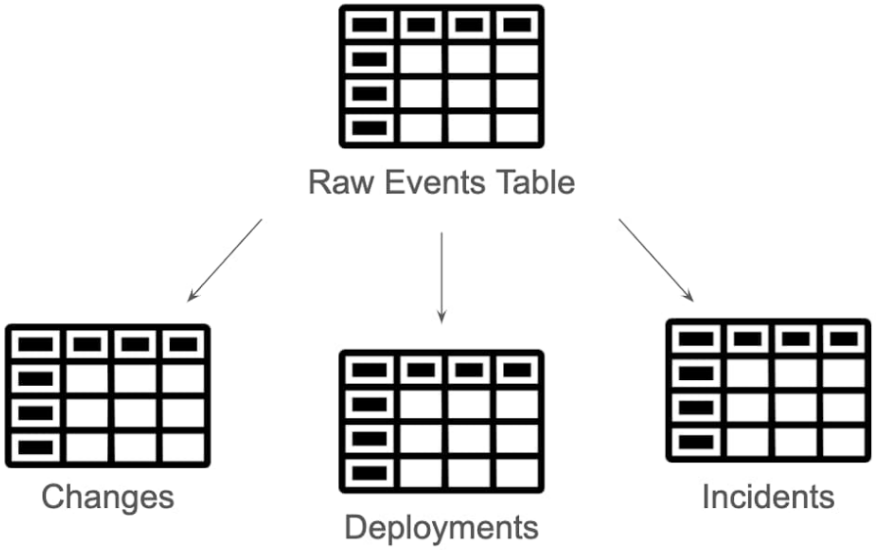
2. El objetivo de Cloud Run es publicar todos los eventos en Pub/Sub.
3. Una instancia de Cloud Run está suscrita a los temas de Pub/Sub, realiza una ligera transformación de datos e ingresa los datos en BigQuery.
4. La vista de BigQuery para completar las transformaciones de datos y alimentar el tablero.



Extracción y transformación de datos

Una vez que los datos sin procesar están en el almacén de datos, hay dos desafíos: extracción y transformación. Para optimizar la flexibilidad empresarial, ambos procesos (extracción y transformación)se realizan con SQL. Four Keys usa consultas programadas de BigQuery para crear las tablas descendentes a partir de la tabla de eventos sin procesar.

Darío Martínez Bajo
Diego García Quirós
Sonia Fernández Coto

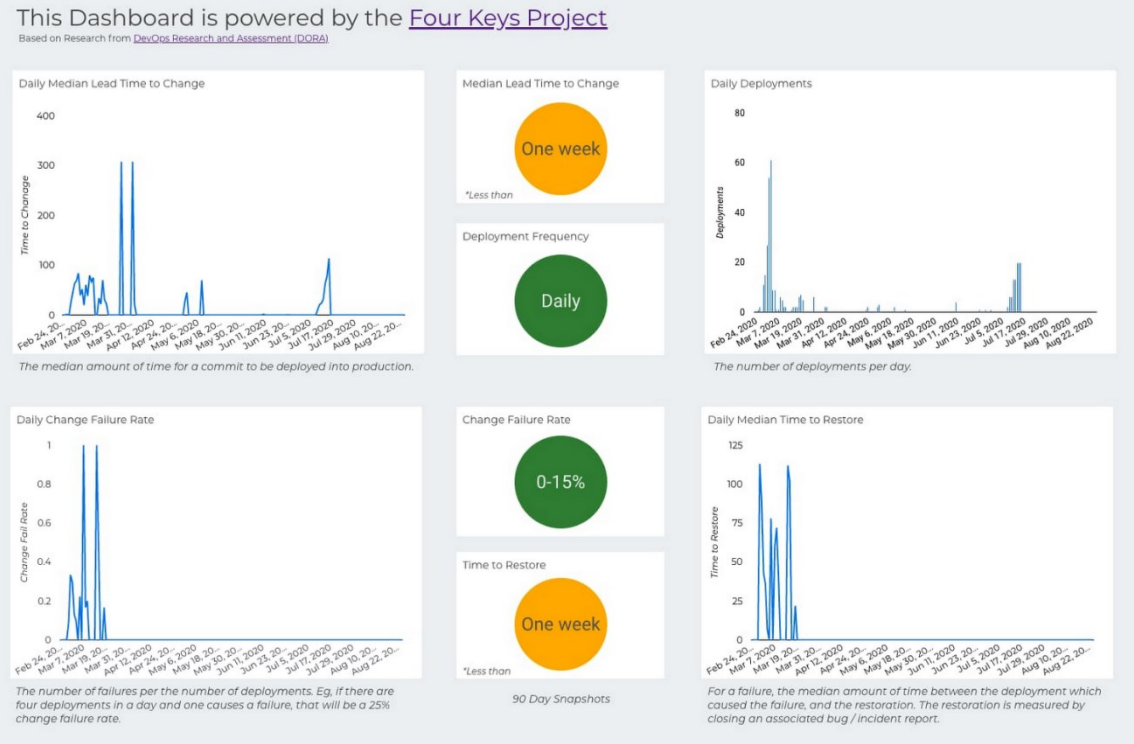


Four Keys categoriza los eventos en Cambios, Implementaciones e Incidentes normalizando y transformando los datos a través de SQL. La definición precisa de un cambio, implementación o incidente depende de los requisitos comerciales de un equipo, por lo que es aún más importante tener una forma flexible de incluir o excluir eventos adicionales.

¡BigQuery incluso te permite escribir funciones JavaScript personalizadas en SQL!

El tablero (dashboard)

Darío Martínez Bajo
Diego García Quirós
Sonia Fernández Coto



Con todos los datos ahora agregados y procesados en BigQuery, puede visualizarlos en el panel de Four Keys. El script de configuración de Four Keys utiliza un conector que permite asociar sus datos a la plantilla del tablero de Four Keys. El tablero está diseñado para para mostrar un registro continuo del desempeño y rendimiento de las Four Keys. Esto permite que los equipos de desarrollo que tengan una idea de una caída en el rendimiento desde el principio para que puedan mitigarlo. Alternativamente, si el rendimiento es bajo, los equipos verán los primeros signos de progreso antes de que se actualicen los diagramas.

Código de colores

El tablero tiene un código de colores para mostrar el rendimiento de cada métrica. El verde es un rendimiento fuerte, el amarillo es un rendimiento moderado y el rojo es un rendimiento deficiente.