



Unidad 1 / Escenario 1

Lectura fundamental

Fundamentos de la Integración Continua

Contenido

- 1 Fundamentos de la Integración continua
- 2 Desafíos de la integración continua
- 3 Importancia de las pruebas automatizadas

Palabras clave: despliegue continuo, calidad, iteración, repositorios, pruebas.

1. Fundamentos de la Integración continua

Durante muchos años, el desarrollo de aplicaciones se había llevado a cabo de una manera muy rudimentaria, de manera manual y poco sincronizada. Era común encontrar equipos de desarrollo poco comunicativos, se veía el desarrollo como una responsabilidad del individuo y no como una integración de varios sistemas. Con el crecimiento tecnológico, la expansión y la acogida mundial de aplicaciones y servicios en Internet, cada vez es necesario reducir el tiempo de producción al máximo, a raíz de eso surge la idea de integración continua.

1.1. ¿Qué es la integración continua?

Martin Fowler uno de los autores más característicos en la aplicación de prácticas ágiles, hace referencia a un equipo de desarrollo que frecuentemente realiza un proceso de integración, para esto define la integración continua como una practica en la que varios desarrolladores actúan de forma constante, y estos, hacen parte del equipo como un conjunto, así, idealmente: *“típicamente cada persona integra al menos una vez al día, generando varias versiones por día. Cada versión ejecutable es verificada por un sistema automático de integración y pruebas para detectar errores de integración lo más rápido posible (Fowler, 2006)”*.

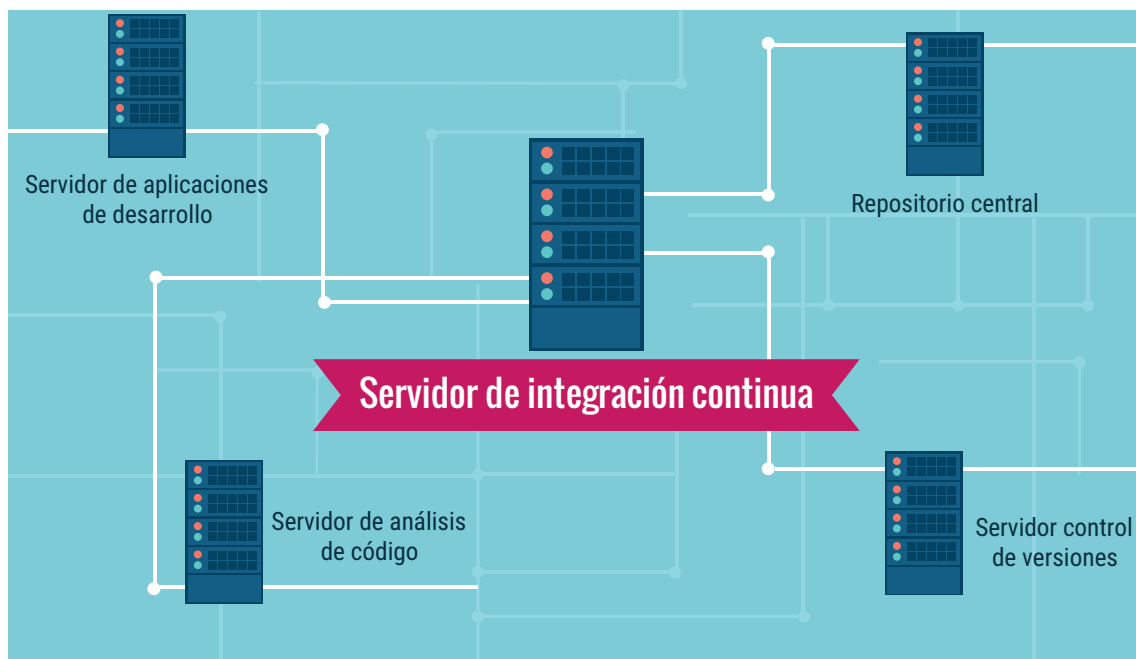


Figura 1. Organización servidores dentro de la Integración Continua

Fuente: Elaboración propia (2017).

Como se muestra en la figura 1, el sistema de integración continua, tiene una organización específica, para el manejo de cada una de sus operaciones, de igual forma un Sistema automático al que hace referencia Fowler (2006) tiene varias características a considerar, para llegar a una integración ideal:

- Alojarse el código Fuente del Software desarrollado dentro de un repositorio; un repositorio al igual que su nombre hace referencia, es un sitio centralizado en el cual se puede guardar y administrar archivos de manera óptima, siendo accesibles por varios miembros del equipo de desarrollo, los interesados del código alojado en él, este término será extendido en la lectura complementaria 2.
- Con un mínimo de comandos, debe ser posible ejecutar la matriz de pruebas de forma automática, dejando a un lado el tiempo de ejecución de pruebas, y reduciendo las fallas de percepción humanas al máximo.
- Facilitar la revisión del código de la manera más simple posible, es decir, cada persona interesada en verificar el buen desarrollo del software, debe tener la capacidad de descargar, compilar y testear de manera automática cada versión que haya sido actualizada en el proceso de desarrollo.
- Garantizar el acceso a la versión más reciente del código mitigando los problemas de comunicación en cuanto a versionamiento, y cada persona trabajara en la última versión, mitigando escenarios, en los que el trabajo realizado sea perdido.

1.2. Diferencias generales entre método tradicional e integración continua

Dentro de las fases de desarrollo de software es necesario el desplegar el resultado del trabajo realizado en este periodo de tiempo, ya sea para iniciar un proceso de pruebas o en algunos casos, para ser desplegado en un ambiente de producción (Ambiente hace referencia a Hardware o Software donde se ejecuta una aplicación) (Laguarda, 2016).

Cómo mejorar...



Mediante incrementos pequeños de desarrollo, tendrá una mejor evaluación de las tareas que ha realizado, esto permite hacer pruebas más eficientes y entregas de calidad.

Ese despliegue se hacía de forma manual, es decir, cada desarrollador realizaba sus tareas, paso a paso y en el momento que creía pertinente, realizaba un lanzamiento de su versión, la cual creía estar terminada, esta versión pasaba a ser en el mejor de los casos evaluada por un tester, tomando un tiempo que dilataba la entrega del proyecto, y en la mayoría de los casos el producto es devuelto al desarrollador, por errores que en su momento no fueron percibidos, esto y algunos otros escenarios generan en los proyectos las características de la tabla 1.

Tabla 1. Características negativas en proyectos convencionales

Características proyectos convencionales
Lento
Costoso
Susceptible de errores humanos
Calidad Media Baja

Fuente: Elaboración propia.

Al ver estos tropiezos constantes en cada desarrollo de software, fue propuesta como solución, la integración continua. Esta metodología propone la automatización de procesos fundamentales los cuales serán nombrados mas adelante, tal vez una característica de gran diferencial es la realimentación constante, realizar pruebas de manera continua con un periodo de tiempo establecido y reducido, permite llegar de una manera natural a una reducción de errores.


1.2.1. Tiempo de iteración y costo

Dentro del desarrollo de software siempre encontraremos procesos que se repiten al paso de un periodo de tiempo, a cada repetición del proceso, se le conoce como iteración, como se observa en la tabla 1, el desarrollo se retrasa como característica debido a que los tiempos se dilatan por las revisiones realizadas esporádicas, al no ser unas realimentaciones activas por parte del tester, los cambios que son hallados al final, generalmente conllevan a un proceso de corrección mayor. El costo incrementa proporcionalmente al tiempo del proyecto; por lo general, cada cambio realizado generalmente no está estimado al principio del proyecto, elevando los presupuestos y de esa forma bajando la calidad de la entrega, generando así, de igual forma, clientes insatisfechos.

1.2.2. Presencia de errores humanos

En todos los ámbitos, aquellas operaciones que impliquen la interacción de las personas son susceptibles a la presencia de fallas humanas, en el desarrollo de software, es muy común observar dichas fallas: En la administración de versiones, en caso de que un desarrollador a la hora de programar, no tenga claro el versionamiento actual del proyecto puede ocasionar la pérdida de horas de producción e incluso días, de ahí la importancia de un buen manejo de repositorios, que ayudados por herramientas como github o bitbucket, las cuales nos ayudan a la administración de los mismos, reducirán casi a cero la posibilidad de cometer errores al desarrollarlo o añadir nuevas funcionalidades al software.

En la figura 2 se evidencia como el control de cambios es mostrado dentro un historial de “commits” los cuales son cambios realizados al proyecto, en este caso, para el software Bitbucket. Todos los programas para administración de repositorios muestran los cambios efectuados al proyecto, otorgando un acceso libre para los desarrolladores asignados al proceso, a este “historial de cambios”, permite minimizar, en gran medida, el error de sobrescribir datos, omitir cambios o funcionalidades en la integración de equipos.





Author	Commit	Message	Date	Builds
EC2 Default U...	0f51845	last changes	2016-11-18	 develop
EC2 Default U...	1b258eb	sept	2016-09-06	 develop
Eduardo Oliver...	b298378	Merged develop into master	2016-05-11	
EC2 Default U...	dbc6b7e	admin config	2016-05-11	
EC2 Default U...	cde0fa0	commit modify post	2016-05-10	
EC2 Default U...	2f2304e	5 de mayo	2016-05-05	
EC2 Default U...	865e1c8	fix facebook	2016-05-02	
EC2 Default U...	2acfd7	final commit before store	2016-05-01	
EC2 Default U...	b2922fe	changes in hhttpd config and chat	2016-04-29	
EC2 Default U...	3136467	Initial commit	2016-04-27	

Figura 2. Historial de cambios Bitbucket

Fuente: Elaboración propia (2017).

De igual forma, dentro de la ejecución de pruebas, el tester puede omitir situaciones poco comunes, y esto puede afectar específicamente a un usuario en el futuro, depende de la calidad y experiencia del tester, el minimizar la cantidad de fallas dadas por alto en la revisión de las versiones, en el caso de que sea automático (Integración continua) el proceso de testeo, estas pruebas no serán olvidadas, y la presencia de falla en un ambiente debidamente testeado, serán mas cercanas a cero.

1.2.3. Comunicación

En el desarrollo de software, para el buen flujo de proyectos, se encuentran dos tipos de comunicación de gran importancia: entre el equipo de desarrollo y con los interesados del proyecto, como el caso de los clientes.

La integración continua ayuda en gran parte a la comunicación del equipo de desarrollo, de manera interna, al tener un registro activo del manejo del versionamiento del software a través de los repositorios, donde cada desarrollador o tester puede tener acceso a un historial de modificaciones, en caso de que un lazo de comunicación entre el equipo sea roto, por cualquier factor (diferencia horaria, olvido, reuniones no realizadas) el historial de actualizaciones será de dominio público, lo que ayuda considerablemente a la buena comunicación del proyecto.

La integración continua sugiere un feedback constante, de forma externa a la compañía, hacia los clientes hace referencia a una participación activa de los stakeholder o interesados del proyecto, cosa que no se evidenciaba en los proyectos convencionales; era fácil encontrar proyectos en los que solo existían dos reuniones junto con el cliente, una al inicio y otra al final, esto, en la mayoría de los casos, había problemas en la entrega del proyecto, la comunicación entre el equipo de desarrollo y el cliente no era lo suficientemente clara, y esto hacía que fuera necesario volver a un punto de desarrollo, a reconfigurar las operaciones.

1.2.4. Calidad

La integración continua ha mejorado la calidad del desarrollo de software a través de los años al reducir los malos hábitos como resultado de la presencia de la automatización. Es evidente que una supervisión continua y automática, junto con una buena comunicación y una reducción de tiempos de desarrollo, aumenta la calidad de los productos considerablemente, la satisfacción de los usuarios es mayor, tanto así que, en este momento, el no hacer uso de herramientas automáticas en procesos de desarrollo, generará un fracaso.

2. Desafíos de la Integración Continua

Todo cambio generará un contraste cultural dentro de las empresas que decidan implementar la integración continua, por lo que es importante observar dos desafíos retadores dentro de esta implementación, los cuales se desarrollan a continuación.

2.1. Cambio cultural

La integración continua necesita una aceptación cultural por parte del equipo de desarrollo y la compañía, debido a que se hace necesario una continua actualización del código, los desarrolladores deben estar abiertos a una revisión diaria, a sugerencias y a un trabajo en equipo, ya que cada uno de ellos aportará al crecimiento individual de sus compañeros, por lo que es importante esta armonía laboral entre los desarrolladores.

De igual forma, es necesario que una persona dentro del equipo de desarrollo esté pendiente y enfocada únicamente en el cumplimiento de objetivos y la aplicación de herramientas de integración continua, además debe estar basada en la investigación y preocupada por la aplicación de los elementos de punta y calidad de la integración continua.

Los desarrolladores deben estar abiertos a una constante realimentación, no solo por parte de su equipo de trabajo sino también por parte del cliente, para quienes es importante una participación activa debido a que él (el cliente) propone los objetivos del proyecto; esto ayuda a que los cambios realizados a través del proyecto sean menos.

2.2. Implementación tecnológica

La búsqueda continua de nuevas tecnologías de integración tiene que ser parte fundamental dentro de un equipo de integración continua, generalmente aquellos equipos de desarrollo que manejan de manera óptima las mejores herramientas de integración, son equipos de desarrollo que obtienen mejores resultados, no solo en calidad, de igual forma reducen el tiempo en el proyecto y los costos.

Es necesario que busque nuevas alternativas de desarrollo, en muchas ocasiones implementamos modelos por nuestra cuenta, los cuales algunas herramientas tecnológicas ya ofrecen, de forma mas madura, la integración continua sugiere el uso de herramientas que le faciliten el trabajo, y que buscan ayudarle a concentrarle en lo que de verdad importa, el desarrollo de un producto de calidad.

3. Importancia de las pruebas automatizadas

Las pruebas manuales hacen referencia a la actividad de verificación por parte de un especialista en pruebas, que está interactuando con un dispositivo ya sea un computador o un celular. A medida del tiempo han sido desarrolladas herramientas automáticas de pruebas, que reducen y facilitan el desarrollo de estas actividades.

¿Sabía qué...?



En la industria, ¿una de las personas más odiadas por los desarrolladores son los testers? Por lo que las pruebas automatizadas mejoran el ambiente laboral dentro del equipo de desarrollo.

Estas herramientas hacen una comparación automática de un valor esperado y un valor obtenido, generando reportes detallados que ayudan a una conclusión fundamentada en pruebas, bastante veraz que permite la acción inmediata a corrección de errores, por parte del equipo de desarrollo.

Las pruebas automatizadas son necesarias en cualquier desarrollo de software debido a:

- Reducen el costo y tiempo de desarrollo debido a que cada escenario negativo en tiempo de pruebas, al ser atendido de forma manual, será un recurso que tendremos que pagar, las pruebas automáticas no aumentan el costo considerablemente y realizan las pruebas en un menor tiempo.
- De forma manual, es muy difícil realizar pruebas en diferentes plataformas, es decir, en desarrollo web, es necesario hacer las mismas pruebas en navegadores como *Chrome*, internet Explorer y Safari, en caso de automatizar las pruebas, generará algoritmos que sean replicables y reproducibles de forma sencilla para cada uno de los escenarios.
- Al ser las pruebas de interacción humana reducidas, no es necesario de la disponibilidad de una persona para realizarlas, el desarrollador se vuelve independiente de su equipo, sabemos que todos desarrollan en horarios en los que otras personas duermen, esto es grandioso ya que no deberán esperar al otro día para saber si todo está correcto.

Como pudimos observar, la integración continua depende de un correcto flujo y sincronía entre el equipo de desarrollo, incluyendo las pruebas dentro de todo el proceso, siendo realizadas dentro de un periodo de tiempo mínimo. Del buen uso de las herramientas que se han proporcionado a medida del tiempo, depende el éxito del desarrollo de software, la calidad y su tiempo de entrega.

La implementación continua, debe ser un requisito de desarrollo para cada proyecto que sea realizado, esto genera una cultura de calidad y comunicación que ayuda a la realización y proyección de cada compañía de tecnología.

En síntesis...

La aplicación de la Integración Continua en la industria del software, no solo da resultados favorables en el tiempo de desarrollo del proyecto sino que también mejora el ambiente laboral del equipo y la calidad de los productos.



Referencias

Fowler. (2006) Continuous integration. Recuperado de: <https://martinfowler.com/articles/continuousIntegration.html>

GeneXus Community Wiki. (2016). Recuperado de: <https://wiki.genexus.com/commwiki/servlet/>

INFORMACIÓN TÉCNICA



FACULTAD DE
**INGENIERÍA, DISEÑO
E INNOVACIÓN**

Módulo: Énfasis Profesional I (Integración continua)

Unidad 1: Principios de Integración Continua

Escenario 1: Fundamentos de la Integración Continua

Autor: Eduardo Enrique Oliveros Acosta

Asesor Pedagógico: Amparo Sastoque Romero

Diseñador Gráfico: Paola Andrea Melo

Asistente: Eveling Peñaranda

Este material pertenece al Politécnico Gran Colombiano. Por ende, es de uso exclusivo de las Instituciones adscritas a la Red Ilumino. Prohibida su reproducción total o parcial.