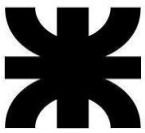


Universidad Tecnológica Nacional

Facultad Regional de Córdoba

Ingeniería en Sistemas de Información

INFORME TÉCNICO

Gestión del Software como Producto – Integración Continua

4K1 – Grupo 5 - 2019

Meles, Silvia Judith (Titular)

Robles, Joaquin Leonel (Ayudante 1ra)

Chiesa María Paula	67395
Frastas Facundo Damián	67394
García Rampini Agustín	67594
Luzara Ezequiel	67788

ÍNDICE

INTRODUCCIÓN

El presente informe técnico tiene como objetivo el desarrollo del concepto de Integración Continua como práctica de desarrollo de software, así como sus características, y su relación con los procesos que le siguen (Entrega Continua y Despliegue Continuo)

La aplicación de la Integración Continua dentro de un grupo de trabajo para el desarrollo de software implica responsabilidad por parte de cada miembro del mismo, así como el conocimiento necesario sobre el manejo de repositorios durante la fase de creación o integración del proceso de publicación de software, ya que implica tanto la utilización de un componente de automatización, como un componente cultural (por ejemplo, aprender a integrar con frecuencia).

Es una técnica que combina de forma continua las actualizaciones en el código llevadas a cabo por cada miembro del grupo de desarrolladores, que evita que las copias locales de un proyecto de un desarrollador afecten al trabajo en general. En la práctica, la integración continua involucra a un servidor centralizado que continuamente hace “pulls” (envío de cambios a un repositorio) en todos los cambios del código fuente, a medida que los desarrolladores hacen “commit” (confirmación de los cambios realizados) y van construyendo el software, notificando así a los demás miembros del equipo si hay alguna falla en el proceso, la cual debe ser solucionada antes de seguir modificando el código.

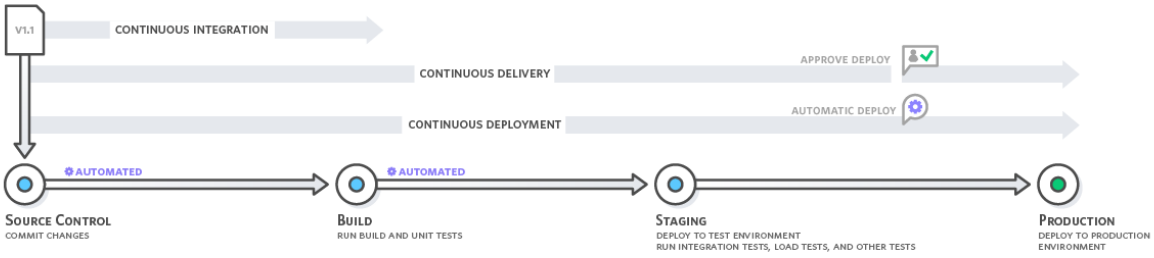
DESARROLLO

La Integración Continua es una práctica de desarrollo de software, a través de la cual los integrantes de un grupo de desarrolladores participan de forma colaborativa, combinando los cambios en el código de un proyecto en un repositorio central, de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. Prevalece el esfuerzo grupal.

El principal objetivo de la aplicación de esta práctica es la de tratar de solucionar errores dentro del código de la forma más rápida posible, a partir de los cambios que se van introduciendo, y a partir de la participación de cada uno de los desarrolladores; así como también tratar de lograr un código más eficiente, más legible, que tenga mejor calidad y que reduzca el tiempo necesario para poder validar la funcionalidad que éste provee.

Para el desarrollo de esta actividad, se actualiza un repositorio compartido con un sistema de control de versiones como Git (ejemplos como GitHub, GitLab, Bitbucket, entre otros), para lo cual, antes de cada envío y actualización en el mismo, se pueden desarrollar pruebas a nivel local para verificar que el código no tiene errores, y así no integrarlo con fallas que en el futuro pueden complicar el trabajo de los demás miembros del equipo.

A su vez, en este repositorio no solo se puede guardar el código de un proyecto, sino que también se pueden almacenar los requerimientos, configuraciones, scripts de testeo, de base de datos, de constructores, etc. Principalmente lo que se hace es generar compilaciones por cada usuario, que van aportando al proyecto de forma colaborativa por cada miembro del equipo, ya que la integración continua se refiere tanto a la fase de creación, como a las pruebas de unidad del proceso de publicación de software. Cada revisión enviada activa automáticamente la creación y las pruebas del código:



Las compilaciones de un proyecto de software se pueden automatizar a partir del uso de algún “CI Server” (Servidor de Integración Continua), considerando que, para esto, es recomendable hacer pequeños registros en el repositorio para que el mismo servidor los controle, y así evitar la dificultad de pasar por alto un error no encontrado. Entre los CI Server más conocidos se pueden encontrar Jenkins, Team Foundation Server (TFS), CruiseControl, Bamboo, etc.

El servidor, al supervisar estos registros, inicia una compilación con cada uno, lo que conlleva que compile el código, o incluso, de forma adicional, calcule la cobertura del mismo, lo limpie, lo minimice, ejecute pruebas unitarias, verifique pautas de estilo, y mucho más. Cuando una de estas compilaciones falla (las condiciones de falla dependen completamente del desarrollador, o del equipo

completo), éste último debe ser notificado, para lo cual, el servidor envía un correo electrónico al programador que cometió el error en el código, o al equipo entero, o a nadie (lo cual no se recomienda por no ser una buena práctica), para luego poder verificar el estado de la misma de vez en cuando.

Esto se realiza para evitar que un desarrollador revise código con errores que ya se encuentra en el repositorio, debido a que éste puede querer realizar cambios para solucionarlo, o agregar más código que depende de éste último, o incluso hacer un "merge" de todos estos cambios; y al darse esta situación, el equipo puede perder el control del sistema en funcionamiento, y sufrir una pérdida de tiempo y esfuerzo al tratar de revertir estos cambios para poder volver a un estado funcional.

Por ello, los CI automáticamente compilan, construyen y prueban cada nueva versión del código que se agrega al repositorio central, asegurándose que el equipo entero esté al tanto, en cualquier momento, de la existencia de código no funcional.

Adicionalmente, el trabajo desarrollado por el Servidor CI debe garantizar un cierto nivel de calidad de software para obtener un software de mayor calidad, por lo que se deberán agregar pruebas para el código compilado. Algunos tipos de estas pruebas son:

- Test Unitario: Prueba unidades de código pequeñas y aisladas;
- Test de Integración: Prueba si el sistema funciona tal como se espera, como el test Big Bang (prueba el sistema cuando están todos los componentes listos);
- Test Incremental: Prueba el sistema con algunos componentes listos;
- Test de Aceptación: Comprueba si la funcionalidad específica (como se describe en la especificación) funciona como se esperaba.
- Test de Humo: Debe probar si las partes más importantes del sistema están funcionando.

Todas las funcionalidad realizadas por el CI descritas son realizadas a partir de scripts de configuración y desarrollo que se escriben de forma colaborativa entre los desarrolladores del proyecto (lo que destaca la importancia del trabajo colaborativo), y a su vez, esto permite la posibilidad de usar dichos scripts para, eventualmente, realizar el despliegue del sistema en entornos de producción con un alto grado de confianza (proceso que se conoce como Despliegue Continuo, y es la etapa que le sigue a la Integración Continua).

Por último, si una compilación en el servidor CI tiene éxito, se crea un artefacto, el cual es un ejecutable del software que debe estar fácilmente disponible para todos en el equipo. Desde que la compilación pasó todos los equipos y todos los criterios de compilación pactadas, este artefacto está listo para ser entregado al cliente.

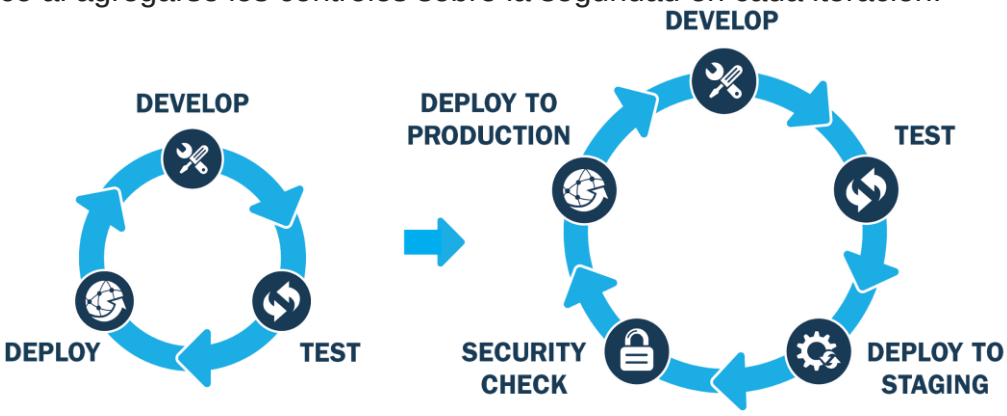
Seguridad en la Integración Continua.

Por lo general, los equipos de desarrollo de software ven a la seguridad de un proyecto como un tema a tratar una vez que el sistema ya ha sido implementado, y está funcionando. Esto es un grave error, ya que la seguridad es algo a tratar durante el propio desarrollo para evitar vulnerabilidades en el código.
Para esto, se debe hacer foco en el tema en las propias iteraciones de desarrollo,

agregando tests de seguridad al proceso de Integración Continua. Al introducir cambios de forma diaria al repositorio central, se consigue un alto nivel de automatización para los atributos referentes a la seguridad, así como también a los atributos funcionales y no funcionales del proyecto, generando de esta forma sistemas más robustos.

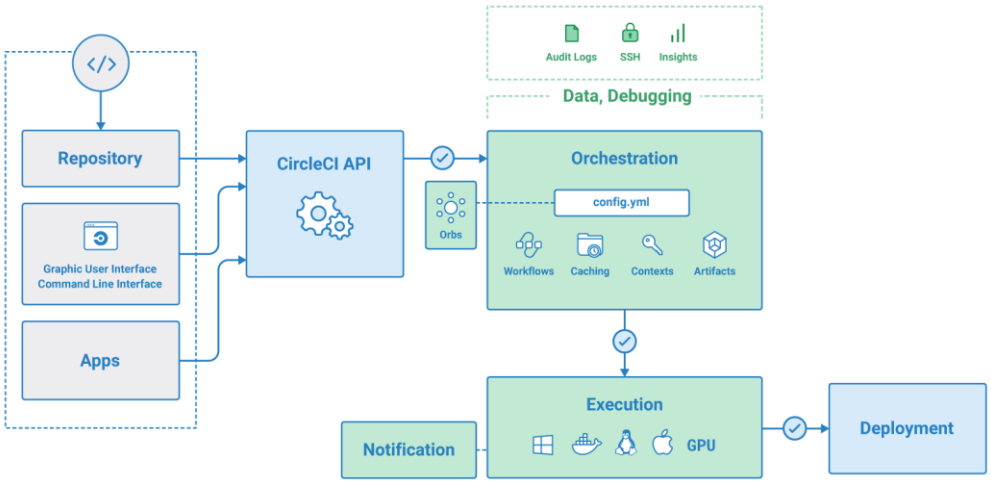
A su vez, al realizar dicha integración se consigue que el software cumpla con las normas de seguridad, así como también se logra la identificación de la seguridad como un atributo de calidad de alta prioridad para el proyecto. Tomando este camino se permite que, desde el comienzo del proyecto, se tomen decisiones inteligentes en aspectos referidos a la arquitectura, diseño y propia implementación, lo que también conlleva a la elección de determinadas tecnologías sobre otras.

Teniendo a la seguridad como un tema de alta importancia, los desarrolladores tienen el incentivo de definir los procesos de desarrollo de una forma que se encuentren orientados a obtener un cierto nivel de seguridad como foco principal para los módulos más importantes del sistema, por ende, el ciclo de trabajo cambia un poco al agregarse los controles sobre la seguridad en cada iteración:



Soporte de Windows para CircleCI.

Un ejemplo de un software Servidor CI es CircleCI, el cual es una plataforma de integración/entrega continua alojada en la nube para Linux, macOS y Android, y actualmente ha anunciado el soporte para Windows también.
Este soporte para Windows le permite a los usuarios ejecutar este CI en una máquina virtual, y actualmente soporta Windows Server 2019, con dependencias tales como .NET, Visual Studio, Windows SDK, Docker para Windows, espacios de trabajo (workspaces) multiplataforma, y cachés.
Algunos de los beneficios de este software para Windows son:
Los trabajos de Windows están basados en VM (Virtual Machine), y proporcionan un aislamiento para el desarrollo completo de las builds.
Usa un ambiente limpio que es creado "just in time" y se destruye una vez que el trabajo finaliza de ejecutarse, para le ejecución de otro nuevo. Esto permite la reproducibilidad de las build, y la seguridad tanto del código como de los datos.
Todas las características de CircleCI como los cachés, los espacios de trabajo (workspaces), trabajos de aceptación y contextos, con el mismo nivel de soporte y de interfaz gráfica, están disponibles para todos los trabajos en Windows.
La plataforma que ahora admite soporte para Windows, se explica a través del siguiente diagrama:



Tanto el código que se va almacenando en el repositorio, como los datos ingresados desde la interfaz de usuario y desde la línea de comandos, como los datos propios de las aplicaciones, pasan por la API del software CircleCI, para que de esta forma se lleve a cabo la depuración (debugging) de los artefactos, como de los contextos, los cachés y los distintos workflows, para llevarlos a la ejecución y generar las consecuentes notificaciones en caso de errores, los cuales, en caso de no existir, permiten que directamente se pueda pasar a la siguiente etapa: el despliegue.

CONCLUSIÓN

La realización de este informe técnico nos permitió como grupo poder aprender el concepto de Integración Continua, el cual era desconocido por todos, como así también entender la importancia que tiene este proceso a la hora de trabajar en un equipo de desarrollo, ya que esto trae varios beneficios tales como mejorar la productividad del equipo (porque permite liberar a los desarrolladores de tareas manuales, y a su vez también fomenta comportamientos que terminan contribuyendo a la reducción de errores de código); localizar los errores más rápido (gracias a las pruebas que se realizan durante el proceso, los errores se pueden encontrar de forma más sencilla y rápida, evitando que éstos se conviertan en bugs más complejos en el futuro que demoren las entregas a los clientes); entregar de forma más regular una versión del software al cliente (al ser un trabajo colaborativo, se desliga a los desarrolladores de tareas manuales, como se mencionó recién, por lo que se permite la entrega de versiones del proyecto de forma más rápida al cliente); ofrecer una mirada actual del software (la integración continua ofrece una mirada en tiempo real del software que se está desarrollado, así como sus medidas de calidad, permitiendo el compromiso inmediato y constante de todos los miembros del equipo a lo largo de todo el ciclo de vida del proyecto); y por último, es una medida de transparencia (esta técnica asegura que todos los stakeholders puedan ver, controlar, comprometerse y contribuir a la evolución del proyecto de software, sin interrumpir al equipo con constantes reuniones para analizar el estado del mismo).

BIBLIOGRAFÍA

Continuous Integration, Delivery and Deployment.
Rossel Sander, Editorial Packt, 2017

Continuous Integration in DevOps.
C. Aaron Cois, Carnegie Mellon University, 2015
https://insights.sei.cmu.edu/devops/2015/04/continuous-integration-in-devops.html

Security In Continuous Integration.
Chris Taschner, Carnegie Mellon University, 2014
https://insights.sei.cmu.edu/sei_blog/2014/12/security-in-continuous-integration.html

¿Qué es la integración continua?
Amazon Web Services
https://aws.amazon.com/es/devops/continuous-integration/

Introducing CircleCI Windows Support, a CI/CD Pipeline on a Windows Virtual Machine
Diogo Carleto, InfoQ, 2019
https://www.infoq.com/news/2019/08/circleci-windows

REFERENCIAS

Cómo redactar un informe técnico.
Ingeniería, Universidad Nacional Autónoma de México
http://www.ingenieria.unam.mx/~especializacion/egreso/Como_redactar_un_informr_tecnico.pdf