# Flujo de trabajos en Github y uso de Docker

Kevin Masis Leandro, Adrian Gómez, Andrés Rojas
Área Académica de Ingeniería en Computadores, TEC
Cartago, Costa Rica
masisleandro@gmail.com
adriangomez2099@gmail.com
anderojasm@gmail.com

**El presente trabajo tiene el propósito de brindar información acerca de los flujos de trabajo en Github, la implementación de la integración continua y el manejo de código en contenedores utilizando Docker.**

## I. INTRODUCCIÓN

La Integración Continua (Entrega Continua o Despliegue Continuo) es una técnica empleada en el ámbito de desarrollo de software para favorecer la comunicación, la colaboración y la integración entre profesionales de operaciones y desarrolladores de Tecnologías de la Información. Los estudiantes de áreas relacionadas con el desarrollo de software como lo es Ingeniería en Computadores necesitan aprender sobre el funcionamiento de CI para el ámbito profesional y laboral.

## II. ANTECEDENTES

### A. Integración Continua

El trabajo más temprano que se conoce e implementa una filosofía de integración continua fue el proyecto *Infuse* el cual; parafraseo, es un ambiente de desarrollo de software experimental que se enfocó en la coordinación de cambios durante las fases de mantenimiento / evolución de proyectos a gran escala, integrando modulo a modulo, a los que se le aplican pruebas unitarias, hasta que finalmente se fusiona con la línea base del proyecto.

### B. Contenedores

La tecnología de contenedores es un método de empaquetar una aplicación para que pueda ejecutarse con dependencias aisladas. Esta técnica surgió aproximadamente en 1970 al ser empleada en la versión 7 de UNIX, y el sistema chroot; este último restringía el exceso de una aplicación a un directorio específico compuesto por directorios raíz y directorios secundarios. Este fue el primer vistazo de un proceso aislado.
Varios años después se implementaron los *Control Groups* o *cgroups* para contabilizar y aislar recursos como *CPU* y *RAM*. Este se usó en el desarrollo de *LXC (LinuX Containers);* alrededor del año 2008, que para la época era la versión más estable y completa relacionada a tecnología de contenedores. Debido a esta estabilidad surgieron otras tecnologías basadas en LXC, como por ejemplo *Warden* en 2011 y más importante aún, *Docker* en 2013.

## III. MODELO INTEGRACIÓN CONTINUA

### A. Ventajas

- Detección temprana de errores.
- Comunicación constante.
- Registro exacto de modificaciones.
- Existencia de una versión para pruebas o demos, funcional y actualizada, si es necesario.

### B. Desventajas

- Cambio de hábitos.
- Requiere servidores y entornos

adicionales.
- Introducción de errores o demoras cuando varios desarrolladores integran su código al mismo tiempo.

## IV. HERRAMIENTAS PARA INTEGRACIÓN CONTINUA

- **Jenkins:** escrito en Java, es de código abierto funciona con distintas herramientas de compilación y sistemas de control de versiones.
- **Travis CI:** herramienta de integración continua, apreciada por su compatibilidad con GitHub, que informa a Travis de los cambios realizados en el código. Existe una versión gratuita del software para proyectos de código abierto.
- **Bamboo:** con el servidor Bamboo, los desarrolladores podrán llevar a cabo la integración, el despliegue y la gestión del lanzamiento continuo. Cuenta con interfaz web y tecnología Java. Ayuda a los desarrolladores mediante la automatización de procesos y funciona con diferentes herramientas de compilación. Existe una versión gratuita para proyectos de código abierto.
- **Gitlab CI:** ofrece un programa propio de integración continua que funciona con la conocida herramienta de control de versiones. Los pipelines pueden configurarse y adaptarse así a los requisitos de cada proyecto. Además, es compatible con GitLab CI Docker.
- **CruiseControl:** este software libre está basado en Java y es compatible con cualquier plataforma. Ofrece a los desarrolladores un panel de control (una página web propia) en el que se puede consultar el estado de los builds.
- **Codeship:** pretende ofrecer a los desarrolladores una opción sencilla para la integración continua. Basándose en la tecnología de contenedores, se pueden crear con él automatismos fácilmente. Para esta tarea, la compañía ofrece dos versiones diferentes: Basic y Pro.
- **TeamCity:** este software comercial pone mucho énfasis en la interoperabilidad con otras herramientas. Su versión estándar es compatible con numerosos programas y el espectro puede ampliarse mediante plugins. Una característica de este software son los Pre-tested commits. TeamCity comprueba el código nuevo antes de integrarlo en la línea principal e informa en caso de errores.

## V. DOCKER

### A. Instalación

El primer paso es ir al sitio web oficial de docker (docker.com) y descargar el instalador de la versión de escritorio para el sistema operativo que desee. Está disponible para Windows, Linux y MacOS.

Se procede a ejecutar el instalador. Este realizará unas descargas previas y luego iniciará con una ventana de configuración.

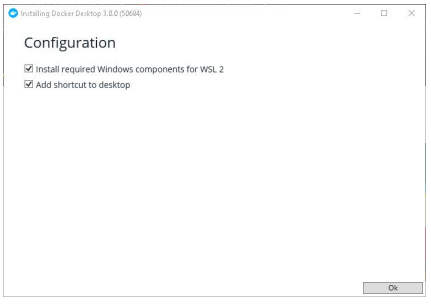


Fig. 1 Ventana inicial docker

El instalador procederá a desempacar paquetes, este proceso puede tardar unos minutos. una vez terminado este proceso pedirá que se reinicie la sesión.

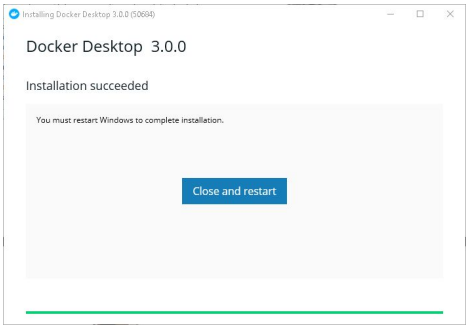

Fig 2 Ventana reinicio de docker

Una vez reiniciado el equipo, Docker comenzará la instalación de ciertas características adicionales y luego estará listo para ser utilizado.

### B. Configuración

El funcionamiento de Docker se basa en el manejo de imágenes y contenedores. Esta metodología permite que el software pueda ejecutarse en cualquier sistema operativo.

Una imagen es una plantilla de un contenedor. Estas pueden almacenar hasta un sistema operativo como Ubuntu o un servidor Apache. Las imágenes se utilizan para crear contenedores. Al momento de crear nuestro contenedor, debemos crear un archivo dockerfile que contenga el nombre de las imágenes que necesita nuestro contenedor para funcionar.

El dockerfile es un documento de texto que contiene todos los comandos que el usuario llama en la línea de comandos para montar la imagen. Docker lee estas instrucciones y las va ejecutando, instala y configura automáticamente el software necesario para que el proyecto funcione.

La creación del contenedor se realiza mediante la línea de comandos de docker. Se selecciona la carpeta que contenga el dockerfile y se construye el contenedor con las imágenes que se añadieron previamente.

### C. Ejemplo

Sin utilizar Docker un posible escenario podría ser el siguiente:

– Kevin tiene en su ordenador instalado Java 8, y está programando una funcionalidad específica de la aplicación con algo que solo está disponible en esa versión de Java.

– Samatha tiene instalada en su máquina Java 7, porque está en otro proyecto trabajando sobre otro código, pero Kevin quiere que Samantha ejecute el código de su aplicación en su máquina. O Samantha instala Java 8, o la aplicación en su máquina fallará.

Este escenario desaparece con Docker. Para ejecutar la aplicación, Kevin crea un contenedor de Docker con la aplicación, la versión 8 de Java y el resto de recursos necesarios, y se lo pasa a Samantha.

Samantha, teniendo Docker instalado en su ordenador, puede ejecutar la aplicación a través del contenedor, sin tener que instalar nada más.

## VI. CONCLUSIONES

La implementación y el uso de integración continua trae grandes beneficios para un desarrollo de software sano y controlado, donde se puedan resolver la mayoría de errores en una etapa temprana, evitando el estrés y caos de integraciones a gran escala donde la depuración de errores sea un trabajo desgastante. Así mismo maximizar la portabilidad y compatibilidad del software con el uso de contenedores.