Entendiendo Integración Continua, Entrega Continua y Despliegue Continuo(CI/CD)

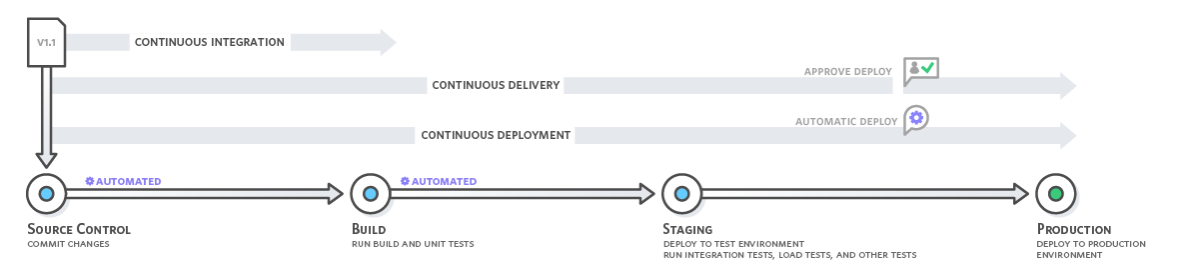


El mundo de los **devOps** es relativamente complejo, pero no hay duda que los beneficios que aporta y el **ROI** (Return On Investment) o retorno de la inversión. Antes que todo necesitamos saber algunos conceptos prácticos.

## Integracion Continua, primer paso

**¿En qué consiste la integración continua?**

Con la integración continua, los desarrolladores envían los cambios de forma periódica a un repositorio compartido con un sistema de control de versiones como Git. Antes de cada envío, los desarrolladores pueden elegir ejecutar pruebas de unidad local en el código como medida de verificación adicional antes de la integración. Un servicio de integración continua crea y ejecuta automáticamente **pruebas de unidad** (Las pruebas unitarias consisten en aislar una parte del código y comprobar que funciona a la perfección. Son pequeños tests que validan el comportamiento de un objeto y la lógica.)en los nuevos cambios realizados en el código para identificar inmediatamente cualquier error. El concepto de CI se introdujo por primera vez hace más de dos décadas para evitar el "infierno de integración", que ocurre cuando la integración se pospone hasta el final de un proyecto.



*La integración continua se refiere a la fase de creación y pruebas de unidad del proceso de publicación de software. Cada revisión enviada activa automáticamente la creación y las pruebas.*

**¿Por qué es necesaria la integración continua?**

Anteriormente, era común que los desarrolladores de un equipo trabajasen aislados durante un largo periodo de tiempo y solo intentasen combinar los cambios en la versión maestra una vez que habían completado el trabajo. Como consecuencia, la combinación de los cambios en el código resultaba difícil y ardua, además de dar lugar a la acumulación de errores durante mucho tiempo que no se corregían.

**Beneficios de la integración continúa**

### Mejore la productividad de desarrolloMejore la productividad de desarrollo

La integración continua mejora la productividad del equipo al liberar a los desarrolladores de las tareas manuales y fomentar comportamientos que ayudan a reducir la cantidad de errores y bugs enviados a los clientes.

### Encuentre y arregle los errores con mayor rapidez

Gracias a la realización de pruebas más frecuentes, el equipo puede descubrir y arreglar los errores antes de que se conviertan en problemas más graves.

### 

### Entregue las actualizaciones con mayor rapidez

La integración continua le permite a su equipo entregar actualizaciones a los clientes con mayor rapidez y frecuencia.

## Entrega Continua, nuestro siguiente paso

La entrega continua, es el siguiente paso luego de CI, es más, lo incluye este proceso, **sin desplegar o llevar a producción el software**, esto quiere decir que (CD) te permite abarcar las siguientes etapas:

## Control de cambios (Source Control): No son más que los commits que subes a tu repositorio

* **Construcción y Pruebas (Build and Tests):** Compilaciones, mover y copiar archivos, crear esquemas de bases de datos, etc, para luego ejecutar pruebas unitarias.
* **Puesta en escena (Staging):** Simplemente desplegar en un ambiente aislado esta versión de software.
* **Producción (Production)**: Esta parte se realiza manualmente

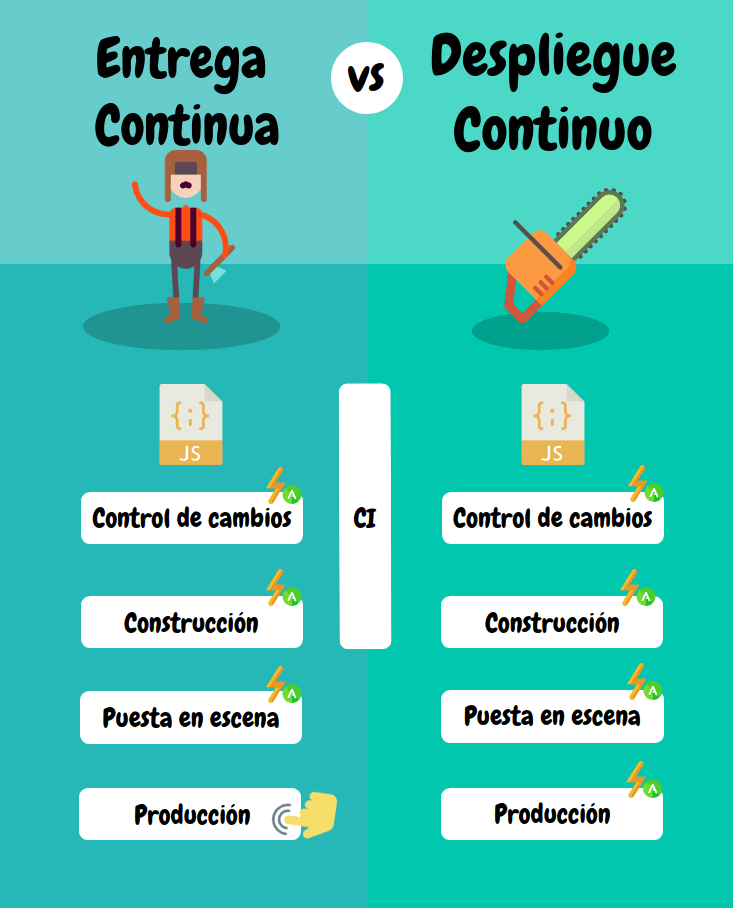
Que no se envíe automáticamente el software a producción no significa que no esté listo, todo lo contrario, simplemente se evita este comportamiento para que el encargado(QA) de probar el software le dé el visto bueno al staging, si todo está ok, se procede a **desplegar a producción a golpe de click**, caso contrario se envía a corregir y repetir el proceso.

## Despliegue Continuo sin demora

Esta es una pequeña variación de la entrega continua, pues si antes se tenía que realizar un despliegue manual, aquí en (CD) simplemente por cada **merge a la rama master** se procede a desplegar automáticamente, el proceso queda de la siguiente manera:

* **Control de cambios (Source Control)**
* **Construcción y Pruebas (Build and Tests)**
* **Puesta en escena (Staging)**
* **Despliege a Producción (Production):** Última versión corriendo en la web principal

Estoy seguro que esta infografía resume muy bien el flujo que anteriormente se mencionó, dale un vistazo.



## Herramientas disponibles

La elección de una u otra opción dependerá siempre de lo que demande tu proyecto, no es lo mismo tener 1 o 2 desarrolladores, que tener un equipo entero a cargo, que corra en un lenguaje tan versátil como Nodejs, el clásico PHP o la bala de Golang, si vas a necesitar o no soporte para Docker, etc. Pero no te preocupes tienes una infinidad de sabores disponibles para elegir, y para hacerlo más fácil aún, estas son las herramientas principales y más populares que puedes encontrar al día de hoy.

* [Codeship](https://codeship.com/): Inicias desde $0
* Resultado de imagen de Circle CI[Circle CI](https://circleci.com/): Inicias gratis
* [Gitlab CI](https://about.gitlab.com/features/gitlab-ci-cd/): Gratis para equipos pequeños
* **Resultado de imagen de travis ci**[**Travis CI**](https://travis-ci.org/): Open source
* ****[**Jenkis**](https://jenkins.io/): Open source
* [Appveyor](https://www.appveyor.com/): Free para proyectos open source
* **Bamboo**
* AWS
* TFS
* Y algunas decenas más

GitHub + Travis CI

GitHub cuenta con varias extensiones que podemos agregar a nuestros repositorios y hoy vamos a hablar de Travis el cual es un servicio de integración continua en la nube, este servicio soporta gran cantidad de lenguajes como Android, C#, Java, Haskell, entre otros.

Si se están preguntando para que me sirve Travis bueno cuando nosotros enviamos un push con los commits éste se encarga de ejecutar un build del proyecto y nos informa del estado del proyecto de manera que podemos enterarnos en cuestión de segundos si nosotros o alguno de nuestros compañeros ha enviado algún o algunos cambios que contienen errores.

A continuación vamos a integrar el servicio con un repositorio de Java empleando Maven para manejar las dependencias por lo que deben de contar con un repositorio con un pom.xml para poder seguir el tutorial o bien pueden hacer un fork de [acá](https://github.com/ccasbre27/Demo-Maven-Gis).

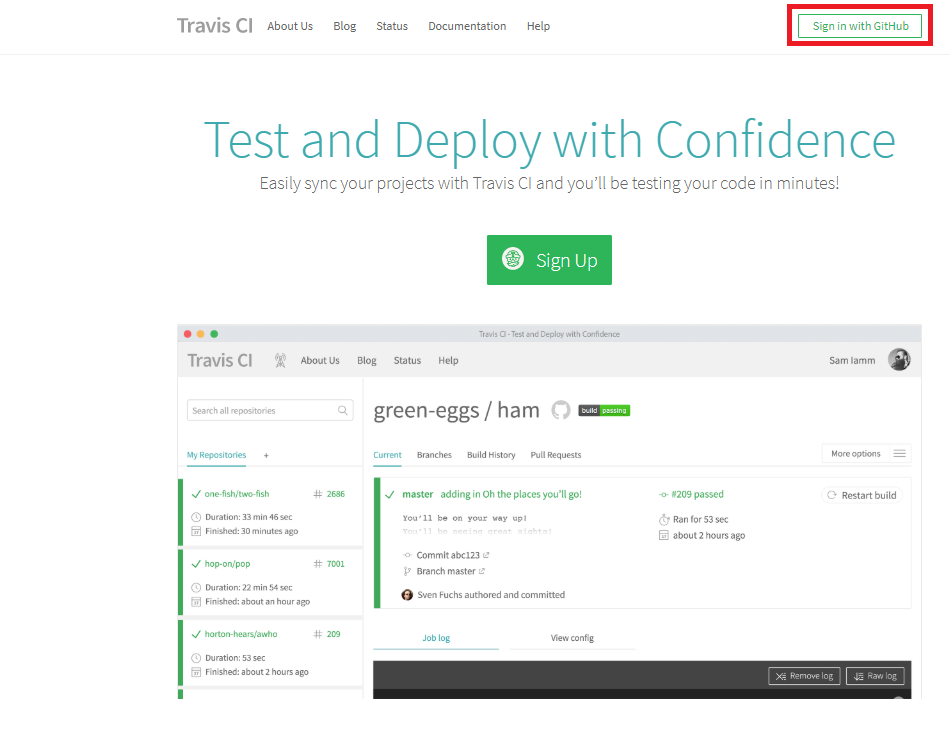
Primero vamos a ingresar en las configuraciones de nuestro repositorio en GitHub.

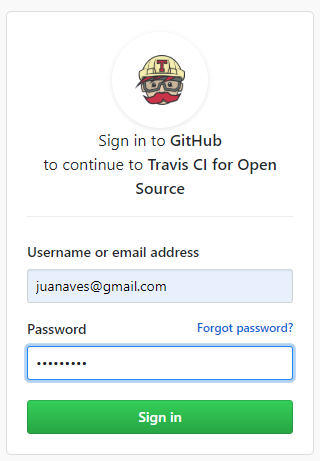
<https://www.juanonlab.com/blog/es/integracion-continua-con-travis>

#### ****Crear una cuenta en Travis****

Lo primero es crear una cuenta en [travis-ci.org](https://travis-ci.org/). Si es un proyecto público de github no es necesario pagar nada ni te va a pedir tarjetas de crédito, ni hay tiempos de prueba.

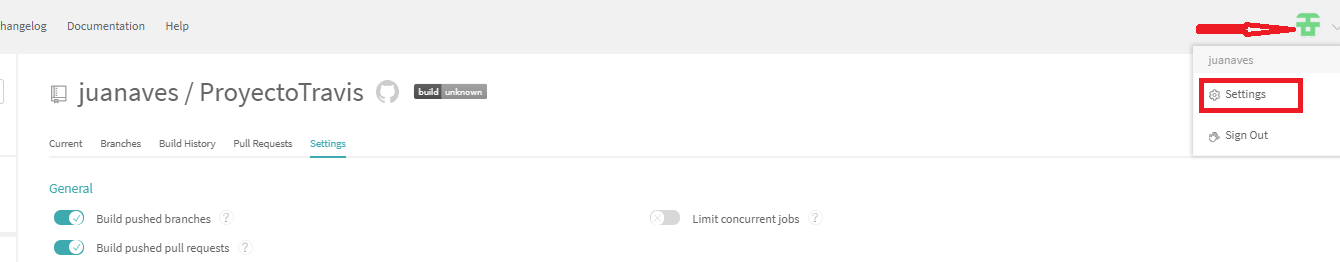
En la página principal como se observa debes utilizar tu cuenta de github. Debes hacer clic en **Sign with GitHub**. Te dirige a una pantalla para insertar tu login y password.



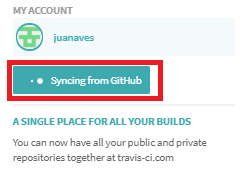


Una vez dentro de la aplicación (<https://travis-ci.org/>) hay que hacer 2 cosas:

1.- Ir a la pantalla de **Settings.**Al hacer clic en tu avatar se despliega un menú donde se encuentra esta opción.



2.- Una vez dentro hace clic en el botón **Sync Account** para que Travis sincronice con tu cuenta de gitHub.



Cuando aparezcan los repositorios se debe deslizar el selector sobre aquellos proyectos donde quieras que trabaje Travis. En la pantalla se observa que sólo está activo ***ProyectoTravis***.



Si tu repositorio no dispone de un fichero yml de travis **NO** va a hacer nada. **Es necesario tener configurado uno**. Por ejemplo el fichero que yo he utilizado es este:

 En el nombre ingresamos “.travis.yml” y en el contenido las siguientes **instrucciones**

language: java

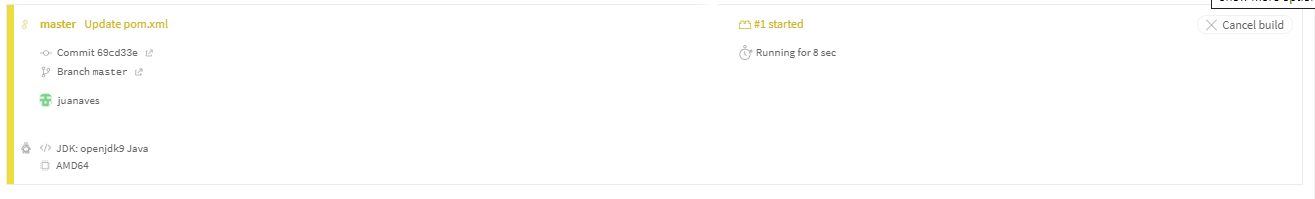
jdk: openjdk9

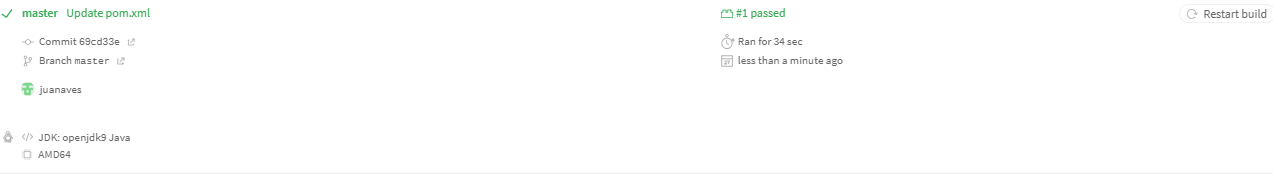
Las **instrucciones anteriores** le indican el lenguaje de programación y la versión del jdk que debe utilizar, se pueden agregar muchas tareas más como enviar un correo electrónico con el estado, compilar con diferentes versiones de un SDK, entre otros.

Se debe ubicar en el **directorio raíz** de tu proyecto.

Existe muchas más opciones de configuración en la página de travis puedes comprobar [otras opciones de configuración](https://docs.travis-ci.com/user/customizing-the-build).

Travis reconocerá al fichero y lo primero que hará será compilar la rama donde esté este fichero (master en este caso). En amarillo cuando está en proceso y verde cuando haya terminado con exito. Rojo si hay algún error. Para acceder a estas pantallas se debe hacer clic en **Dashboard** y luego clic en en el repositorio (optionalJava).

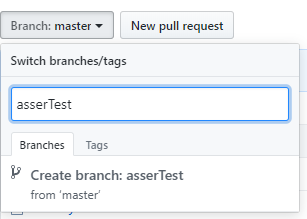


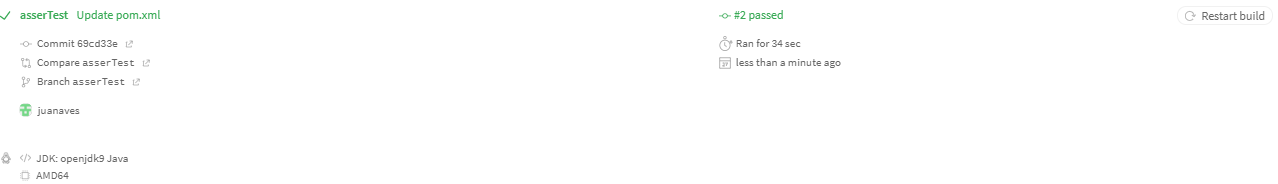


En caso que haya errores aparece algo similar a lo siguiente.

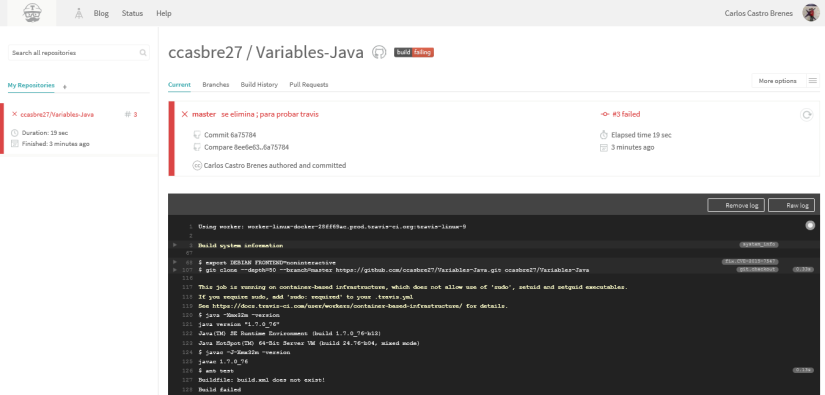
La integración con github es perfecta y desde github se puede comprobar cómo trabaja travis.

Voy a crear una rama nueva llamada ***asserTest***y cambiar el test que tengo **para que fallé.** Subiré la rama a github y se podrá comprobar el estado de la misma desde github.





Desde github se incorporan unos iconos que indican si la rama se ha construido de manera correcta o no. Si hacemos clic en el icono se muestra un mensaje de Travis indicando que algo no ha funcionando bien.



Bueno **Travis CI** es una herramienta muy útil para el desarrollo de software que pertenece a las buenas prácticas de **DevOps** y su integración con GitHub es sencilla.