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?**

Hay una práctica que, por desgracia, sigue siendo una gran desconocida cuando hablo con gente que se dedica al software. Me estoy refiriendo a la integración continua (CI).

Si bien es cierto, que las pruebas unitarias tienen una funcionalidad clara, no valen de nada si **no las ejecutas**. Esto puede pasar por muchas razones, pero una de ellas, es que trabajas sobre una funcionalidad que hay que pasar a la rama principal urgentemente y no hay tiempo de lanzar todas las pruebas unitarias. Durante el desarrollo de esa característica, compruebas que todo funciona bien, y subes tus cambios, tranquilo, hasta que de repente… ¡¡Algo se ha roto en otro sitio!!

**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

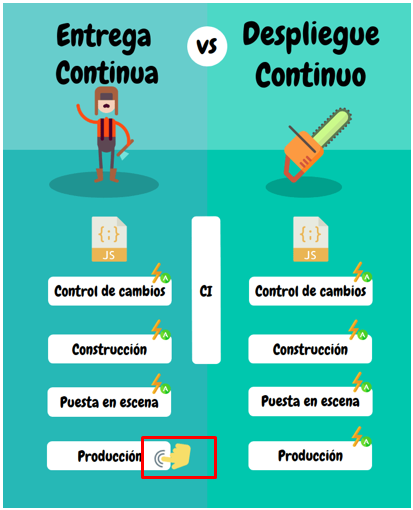
La **entrega continua**, es el siguiente paso luego de CI, es más, lo incluye este proceso, **sin desplegar o llevar a producción el software**

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**,

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

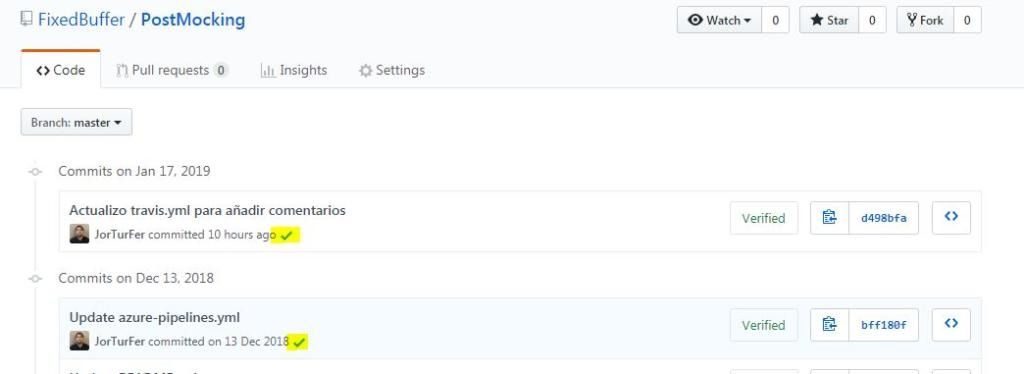
Gracias a herramientas como [AppVeyor](https://www.appveyor.com/" \t "_blank), [Travis CI](https://travis-ci.org/) o [Azure Pipelines](https://azure.microsoft.com/es-es/services/devops/pipelines/" \t "_blank), (**los 3 son gratis si nuestro repositorio es público**) podemos conseguir **automatizar este trabajo**, de modo que **no** tengamos que **preocuparnos** de compilar y ejecutar las pruebas. Además, estos sistemas nos aportan otra ventaja adicional, podemos compilar y ejecutar pruebas en diferentes entornos, como veremos a lo largo de esta entrada sobre Travis, y la siguiente que será sobre [Azure Pipelines](https://www.fixedbuffer.com/index.php/tag/azure-pipelines/) .

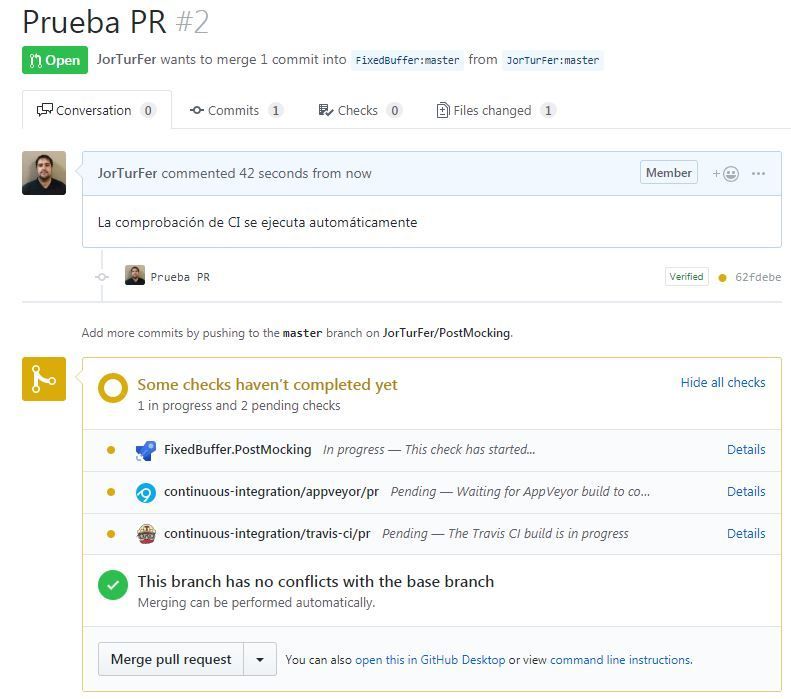
GitHub + Travis CI



## Funcionamiento de los servicios CI

En primer lugar, podemos preguntarnos cómo funcionan estos sistemas de CI. Si utilizas GitHub es muy simple, están totalmente integrados, y con unos pocos clicks podemos hacer que cada vez que hagamos un push al repositorio (o nos manden un Pull Request), automáticamente se lance la compilación y ejecución de las pruebas del código, sin tener que preocuparnos de nada más. Además, estos servicios suelen trabajar usando un fichero de configuración “**.yml**”, en el cual definimos todos los pasos que queremos ejecutar, de modo que es fácil reutilizar las configuraciones en nuestros diferentes proyectos con unos cambios mínimos. Una vez terminen, veremos en el historial de commits un indicador de si la integración ha ido bien o no, al igual que en los PR:





Además, estos servicios suelen permitirnos utilizar **badges** que podemos añadir a nuestro **readme.md** para saber siempre a simple vista el estado del proyecto:

## Integración continua con Travis

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.

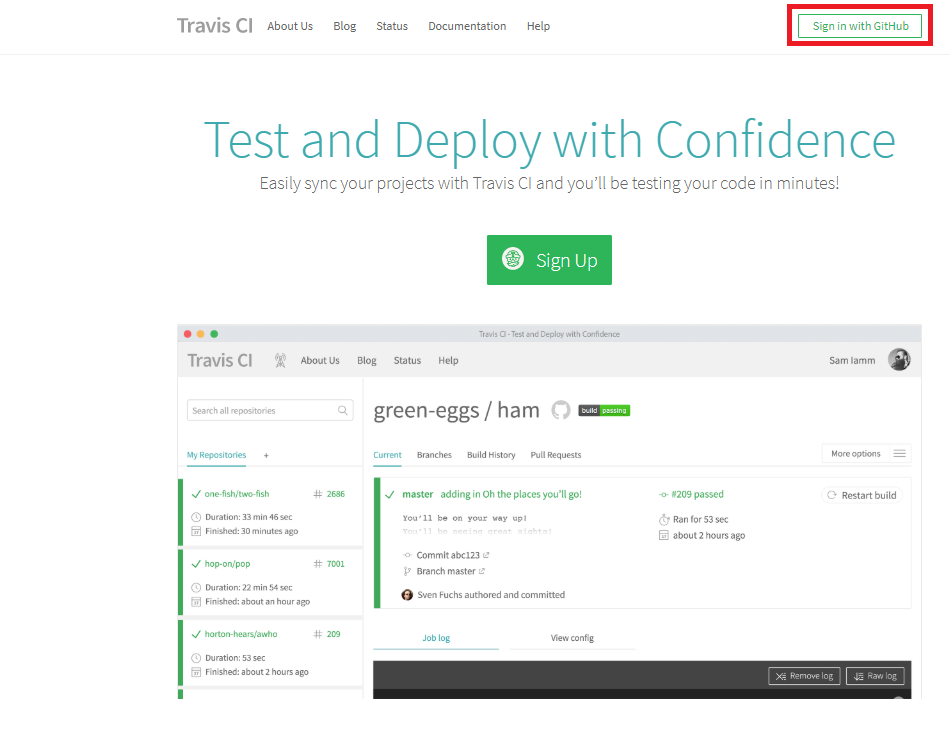
Si se están preguntando para que me sirve Travis bueno cuando nosotros enviamos un push con los commits éste se encarga de ejecutar un build del proyecto y nos informa del estado del proyecto de manera que podemos enterarnos en cuestión de segundos si nosotros o alguno de nuestros compañeros ha enviado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

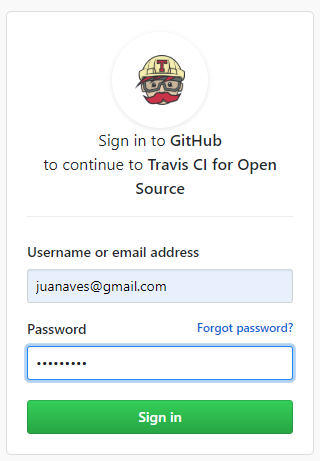
A continuación vamos a integrar el servicio con un repositorio de Java empleando Maven para manejar las dependencias por lo que deben de contar con un repositorio con un pom.xml para poder seguir el tutorial o bien pueden hacer un fork de [acá](https://github.com/ccasbre27/Demo-Maven-Gis).

#### ****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.

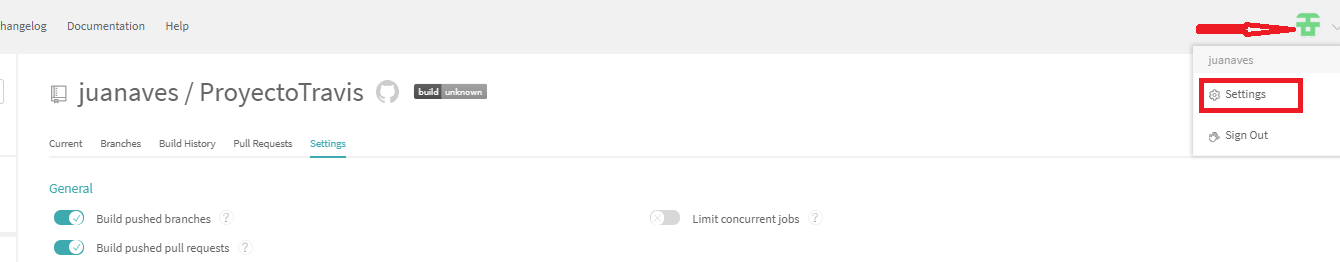
En la página principal como se observa debes utilizar tu cuenta de github. Debes hacer clic en **Sign with GitHub**. Te dirige a una pantalla para insertar tu login y password.



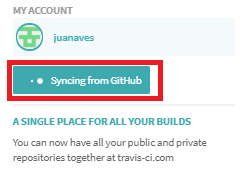


Una vez dentro de la aplicación (<https://travis-ci.org/>) hay que hacer 2 cosas:

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

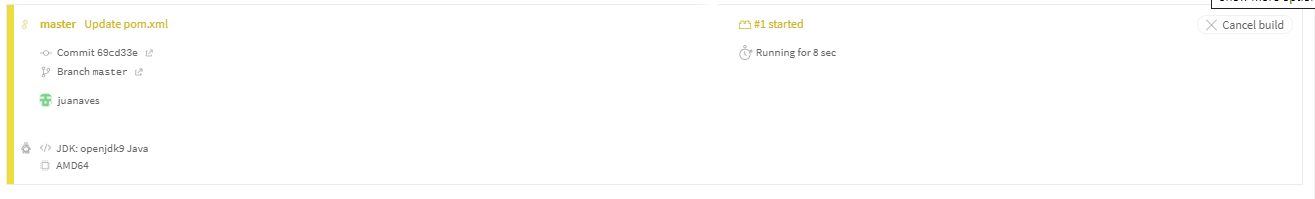
jdk: openjdk9

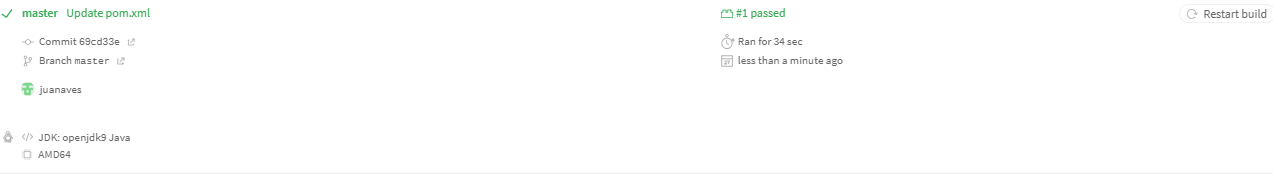
Las **instrucciones anteriores** le indican el lenguaje de programación y la versión del jdk que debe utilizar, se pueden agregar muchas tareas más como enviar un correo electrónico con el estado, compilar con diferentes versiones de un SDK, entre otros.

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).

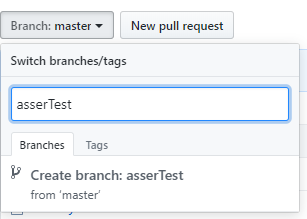


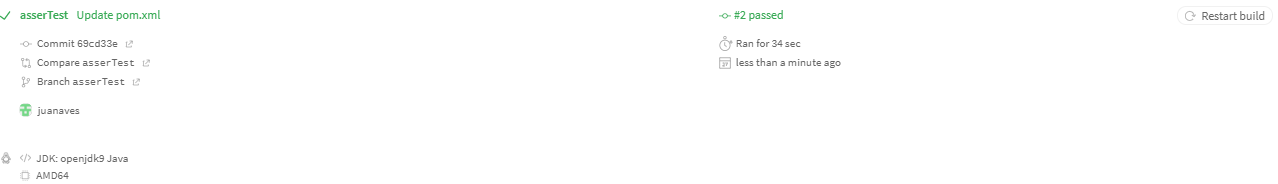


En caso que haya errores aparece algo similar a lo siguiente.

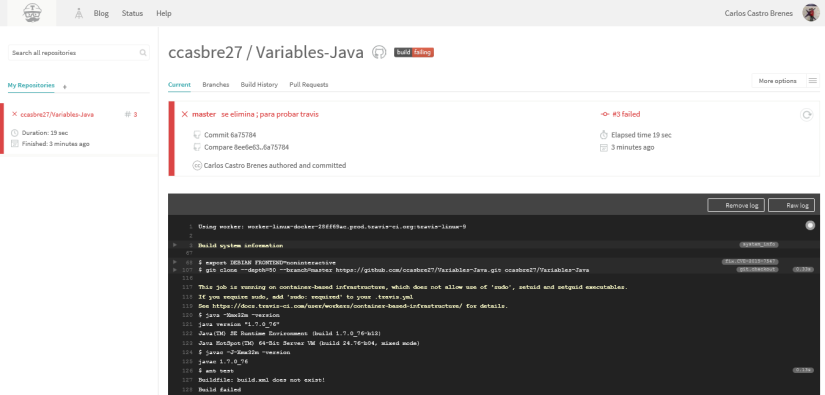
La integración con github es perfecta y desde github se puede comprobar cómo trabaja travis.

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.

### La configuración del archivo .travis.yml se puede ampliar con varios parámetros por ejemplo

language: java

jdk:

- oraclejdk8

notifications:

email:

- integrante1@unsam.edu.ar

- integrante2@unsam.edu.ar

Reemplazá integranteX@unsam.edu.ar por los mails de los integrantes del grupo e incluí a tu tutor.

Si queréis hacer algunos ajustes extras en las notificaciones, podéis buscar [la documentación oficial de Travis](https://docs.travis-ci.com/user/notifications/#Configuring-email-notifications).

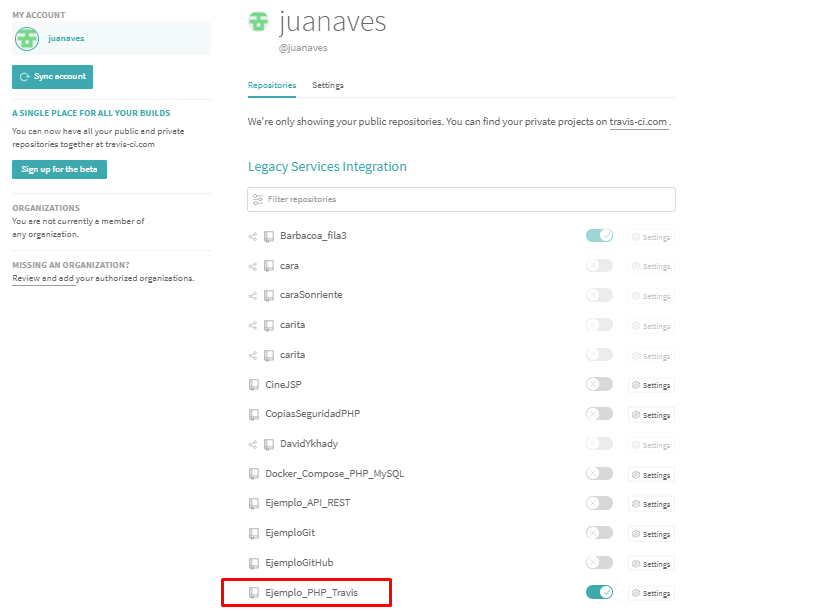
#### **Qué pasa entonces**

A partir de aquí, **cada vez que hagas** un **push**, [Travis](https://www.travis-ci.com/) como servidor de integración continua

* clonará el repositorio tuyo
* lo compilará en la tecnología en la que trabajaste
* ejecutará los tests
* y generará el nuevo status de tu proyecto
* en caso de error, te mandará un mail avisándote que el build falló
* y si anduvo ok, te dirá que el build pasó
* de aquí en más, cada vez que hagas un push al repositorio, si anda ok no te dirá nada, pero si falla, te mandará un mail. Es mejor saberlo antes, no?

Vamos a ver otro ejemplo utilizando php

Sincronizamos nuestra cuenta para que se nos muestren los repositorios, y simplemente, activamos el repositorio que nos interesa:



Con esto, ya hemos indicado a Travis que queremos que lance el CI cada vez que detecte cambios en el repositorio, y Travis se encarga de configurarlo todo en GitHub para que le notifique esos cambios. A partir de este momento, cada vez que hagamos un push al remoto, se iniciara una compilación. En este momento, fallará al no tener el «.yml», asi que vamos a añadirlo.

Fichero .travis.yml

Vamos a crear un fichero en el repositorio que se llamará .travis.yml (el primer «.» delante de travis hay que ponerlo también). En el vamos a poner lo siguiente:

language: php

php:

  - "7.0"

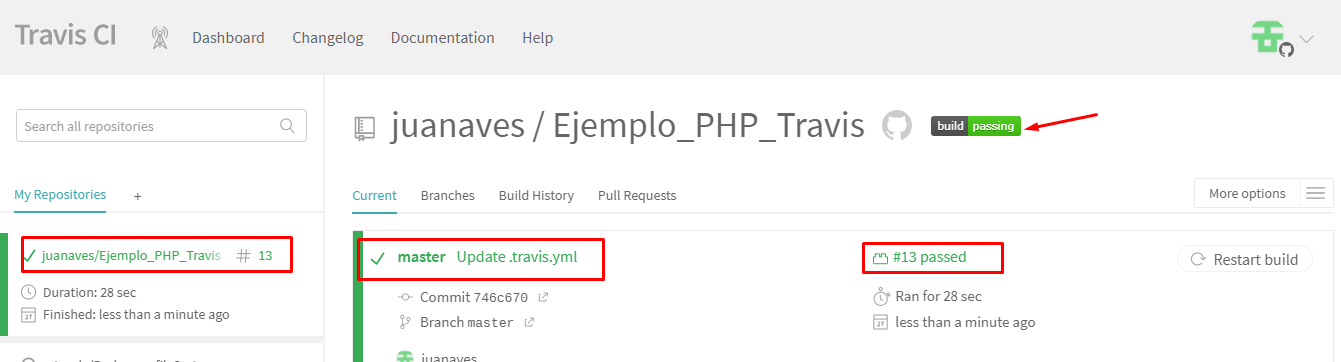
 # list any PHP version you want to test against

script:

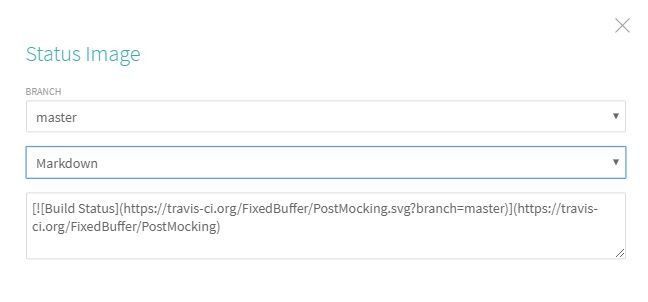
- php index.php

Si nos fijamos, simplemente le estamos indicando al servicio que es lo que queremos que haga. Le indicamos el lenguaje, le indicamos la versión del SDK, y los sistemas operativos sobre los que trabajar.

Si todo va bien, una vez que termine, si vamos a **travis**, veremos algo como esto:



Como comentaba antes, Travis nos permite generar badges para saber el estado del repo, esto se hace tan fácilmente como **pulsando sobre** el **badge**, y nos mostrara una ventana para seleccionar donde lo queremos poner. Simplemente, tenemos que seleccionar donde lo vamos a colocar, y nos dará el código copy-paste.



Con esto, ya tenemos una primera aproximación a la Integración Continua con Travis CI. En las siguientes entradas de esta serie sobre CI/CD, veremos [Azure Pipelines](https://www.fixedbuffer.com/index.php/tag/azure-pipelines/), y hablaremos sobre el **despliegue** **continuo** (CD), que es tan facil como el CI, y nos permite por ejemplo, automatizar la publicación de paquetes nuget, webs, subir los binarios a un FTP, etc…