Introducción

Uno de los problemas más frecuentes a la hora de trabajar en equipo es la sincronización del trabajo realizado entre los distintos miembros. Si las personas no se sincronizan de forma correcta se pueden producir discrepancias y problemas adicionales a la hora de juntar codigo proveniente de varias fuentes. Para solucionar este concepto se diseño la Integración Continua, un sistema que en cada momento que se añada código fuente a un repositorio, una serie de tests son evaluados sobre el estado actual para realizar un informe de si ha habido problemas al añadir el codigo o no.

Aunque este concepto fue diseñado principalmente para entornos profesionales con equipos de trabajo y personal no estrictamente organizado, puede aplicarse de forma similar a otras tareas, como en este caso la supervisión del repositorio de una asignatura, de forma que se puedan hacer comprobaciones sobre los alumnos automaticamente y no sea necesario el trabajo manual de alguno de los profesores.

Conceptos Básicos

**Integración Continua**: Sistema por el cual se realizan unas pruebas cada vez que se actualizan los ficheros de un proyecto para asegurar la estabilidad del mismo

**Travis:** Sistema externo que se encarga de realizar integración continua en proyectos almacenados en GitHUB

**Build:** Conjunto de ficheros del proyecto que determinan su estado en un determinado momento.

**Test:** Pruebas que se realizan para determinar si la build es correcta o contiene errores

**Commit**: Cambios realizados en el repositorio y validados

**Push:** Envio de los commits realizados a un repositorio remoto. En el caso de GitHUB, realización de commits directamente mediante la interfaz web.

**Pull-request:** Petición de que se incluyan los commits de un repositorio clonado en el original

**Webhook:** Petición HTTP a una URL para provocar un comportamiento especifico. En este contexto se envia cuando la build es terminada y es obtenido para la realización de comentarios y otras acciones.

TODO

Funcionamiento General

## Elección de heramienta de integración continua

A la hora de implementar un sistema de integración continua, la primera decision que hay que tomar es que herramienta se va a utilizar, ya que dependiendo de esta la infraestructura o diseño necesario varia. Se pueden diferenciar dos variantes de forma inmediata:

* Uso de un servidor privado en la compañía
* Uso de un servidor externo como SaaS

La forma de implementar cada variante es diferente, pero para el desarrollo de este trabajo se va a usar Travis-CI.org, una alternativa gratuita en la nube de facil acceso.

## Estructura del repositorio de GitHUB

Por la parte de GitHUB, es necesario definir la organización que tendrá la asignatura. En el pasado esto era realizado con un único repositorio dividido en sub-carpetas. Aunque también se podria realizar de esa forma con integración continua, existe una mejor distribución que facilita ciertas tareas: Crear una organización para la asignatura y separar cada Tarea en un repositorio diferente. De esta forma todo el trabajo queda mucho mas organizado tanto para los profesores como los alumnos, y se facilitan las tareas de integración continua. La creación de organizaciones públicas es gratuita, y los repositorios creados pueden ser tanto públicos sin coste adicional, como privados (siendo necesario actualizar a la versión de pago).

La inclusión de los alumnos se realiza simplemente compartiendo el enlace a la organización, donde aunque no serán miembros, podran hacer cambios como *third-party collaborators* enviando pull-requests*.* Los miembros de la organización seran los profesores y aquellas entidades relevantes, que dispondran de permisos de push a los repositorios directamente, facilitando la realización de cambios.

Esta implementación permite:

* Tanto la organización como los repositorios es pública, permitiendo compartir los conocimientos obtenidos y desarrollos realizados
* Facilitar la modificación de las tareas de los profesores tanto separando cada trabajo en un repositorio diferente, como la realización de cambios a los mismos.
* Limitación de los alumnos a realizar únicamente pull-requests, forzando a que aprendan a manejarse con GitHUB desde el principio.

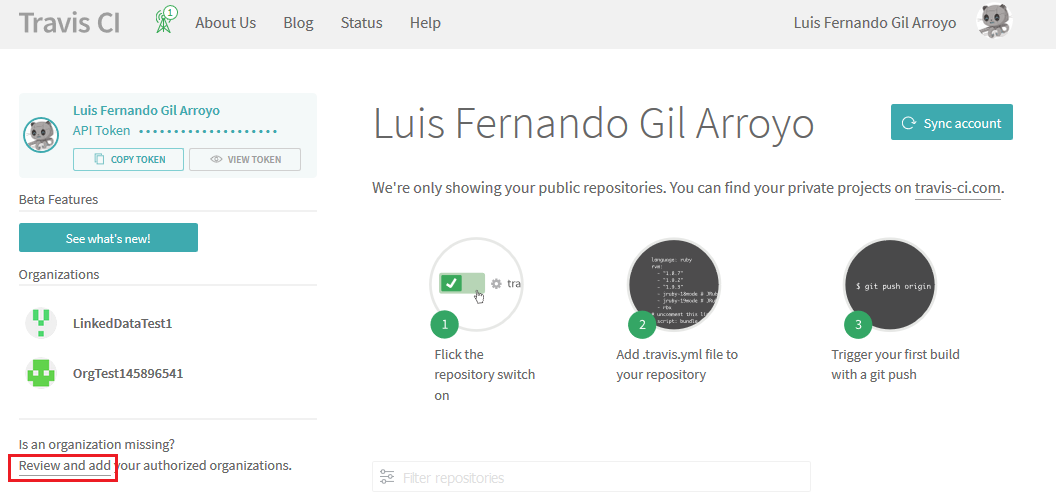
## Repositorios para las tareas y ficheros de build

Una vez creada la organización, es necesario crear los repositorios para el desarrollo de las tareas. Independientemente del trabajo, los repositorios contendran estos minimos archivos:

* .**travis.yml**: Archivo de configuración para el montaje de la build en travis. Una explicación mas detallada de este archivo puede encontrarse en una sección posterior. TODO
* **travis-buddy-failure-template.md y travis-buddy-success-template.md**: Archivos con la plantilla para la realización de los comentarios por la aplicación third-party comentada más adelante.
* **Build/** : Directorio con los archivos bash de prueba especificos de cada tarea

## Autorizar acceso al repositorio

Una vez que estos archivos han sido creados, probados, y configurados correctamente, se procede a acceder a la web [travis-ci.org](https://travis-ci.org/), alli hacer login con una cuenta administradora de la organización y dar permisos de acceso de Travis a la organización como en las siguientes figuras:

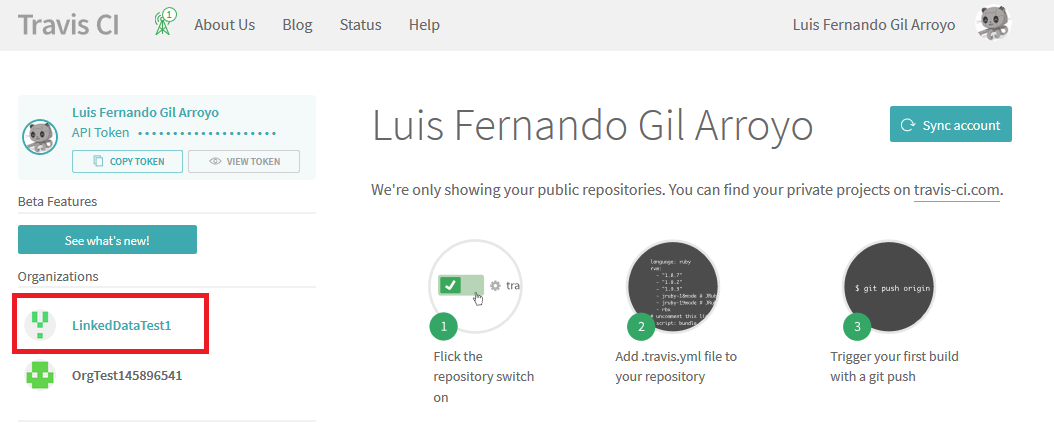


*Figura: Botón para acceder al panel de acceso a las organizaciones del usuario*

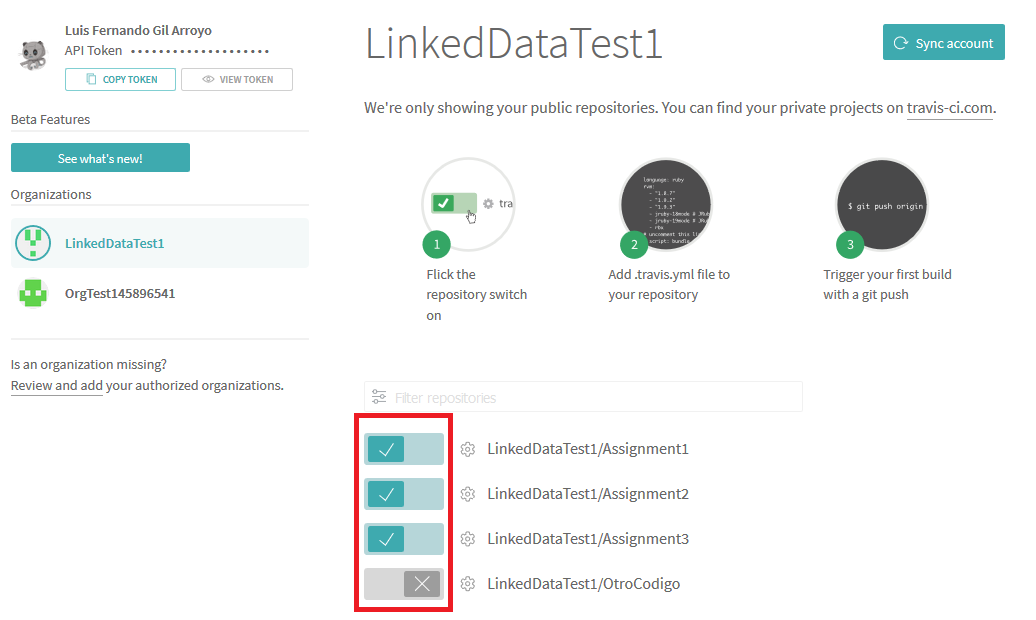


*Figura: Botones de acceso especifico a cada una de las organizaciones del usuario desde GitHUB*

Los permisos puede que tarden unos minutos en verse reflejados en la website de travis, pero una vez que lo hayan hecho, activar los interruptores especificos de cada tarea:

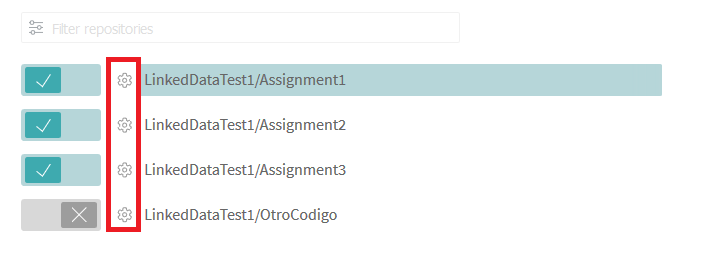


*Figura: Botón de acceso a la configuración Travis de la organización*

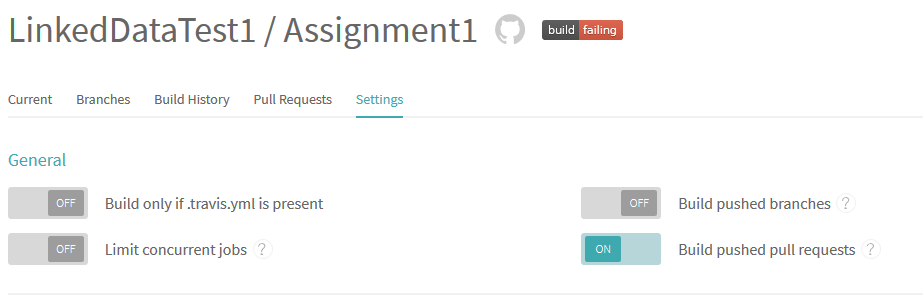


*Figura: Botones para activar y desactivar las builds de cada repositorio de la organización*

En la parte de configuración (settings) de cada uno de las tareas se recomienda desactivar las casillas de “Build on push”, de esta forma los profesores podran hacer push directamente a los repositorios sin tener que preocuparse de lanzar builds sin querer y producir comentarios o errores.



*Figura: Botón de acceso a la configuración Travis de cada repositorio*



*Figura: Botones para activar y desactivar builds en push y en pull-request*

Una vez todo esto esta configurado el repositorio esta listo para su uso.

Servicio integrado externo para la realización de comentarios: TravisBuddy

Para realizar acciones como publicar comentarios es necesario la utilización de la API de GitHub. Para esto es necesario identificarse con un token de usuario, que debido a la naturaleza de las pruebas deberia ser escrito directamente en el codgio creando un potencial problema de seguridad. No es posible hacer uso de variables de entorno en la plataforma de Travis ya que por razones de seguridad son desactivadas para pull-requests realizadas de forma externa al repositorio. Por esto he tomado la decisión de usar el servicio de [TravisBuddy](https://github.com/bluzi/travis-buddy). Este servicio recibe un webhook al finalizar la build y realiza el comentario en la pull-request siguiendo la plantilla que se encuentra en el mismo repositorio, o una por defecto si no existe, existiendo la posibilidad de habilitar y deshabilitar los comentarios añadiendo o eliminando la opción de notificación en el archivo .travis.yml. Mas detalles en la sección de este archivo (TODO). Ya que este servicio es externo no se producen problemas de seguridad ya que los tokens de acceso no son expuestos al publico.

Si se opta por no usar un servicio *third-party* existe la posibilidad de realizar un simple servidor web que reciba las peticiones del webhook y las responda, aunque esto supondria la creación de este código más la infraestructura de despliegue del servidor.

Descripción de las Tareas

## Assignment 1

Este assignment sirve como iniciación del alumno en github y para que se familiarize con las herramientas necesarias.

La anterior estructura consistia de un fichero csv donde el alumno introducía una serie de campos en función de un dataset que él escogía, y realizaba los cambios mediante un push directo.

Esta tarea se remodelaría para que cada alumno cree su propio documento y lo suba mediante una pull-request, quitando de esta forma el paso de realizar push ya que es trivial. Este fichero que el alumno subiría seguiría una determinada estructura de nombre {usuarioGithub}.csv y consistiria de una columna con dos valores, el mismo nombre de usuario de github en la primera y el número de matricula en la segunda. Este fichero es utilizado por otras pruebas para consultar el número de matrícula del alumno, además como está vinculado al alumno podría utilizarse como un archivo de registro automático de las tareas que va subiendo el alumno y si las builds han pasado con éxito o no, y la posibilidad de incluir más campos por ejemplo calificaciones o comentarios del profesor al alumno.

El test realizado sobre este fichero comprueba que este existe (es decir, que tenga ese nombre) y que el CSV esté bien formado, en este caso inicial que tengo dos campos.

Scripts utilizados:(TODO)

## Assigment 2

Esta tarea consiste en la creación de un vocabulario simple en RDF y en TTL, así como subir una imagen del gráfico de los mismos datos.

Estos archivos se subirán en el root del mismo repositorio, y mantendrán la misma nomenclatura que en el pasado {usuarioGithub}.[png/rdf/ttl].

En este caso, al ser los tests automáticos las posibilidades de hacer pruebas sobre estos ficheros aumentan de forma exponencial. Los test realizados incluyen, aparte de comprobar que los ficheros existen, ejecutar un jar con un programa de jena sobre estos ficheros.

Este jar que está pre-compilado (ya que lo único que cambia es el argumento del fichero a comprobar se ahorra el tiempo de compilación y empaquetación del mismo) se ejecuta sobre los ficheros, y comprueba que la sintaxis es correcta, o en caso contrario devuelve un mensaje de error. Aunque actualmente solo tiene este caso de prueba tan sencillo, se puede ampliar de forma simple a realizar otras pruebas, como búsquedas, comprobaciones exactas, etc

Scripts utilizados:(TODO)

## Assigment 3

En esta tarea el alumno es encargado de realizar un codigo en java de ciertas funciones para trabajar con rdf, usando Jena. En este caso es necesario realizar una compilación del código, que es realizada mediante Maven incluyendo en el repositorio el respectivo fichero pom.xml.

La build que se realiza compila el proyecto en un jar junto con los archivos de prueba JUnit y procede a ejecutarlo para comprobar los resultados del código del alumno.

Al igual que en la anterior tarea, la complejidad de las pruebas realizada puede ser aumentada si es requerido.

Scripts utilizados:(TODO)