

CONTINUOUS INTEGRATION

*Introduzione e Descrizione delle
principali funzioni*

*A cura di : Matteo Ballocco & Marco Ena
07/02/18*

• **Continuous integration**

In inglese **Continuous integration** (CI), è una pratica dell'ingegneria del software che implementa processi continui di verifica e controllo di qualità, applicati con frequenza in fase di sviluppo del software.

L'obiettivo è quello di migliorare la qualità del software, e di ridurre al contempo il tempo di consegna dello stesso, sostituendo la tradizionale pratica di applicare il controllo di qualità del software alla fine dell'intero ciclo di sviluppo.

Deployment

Con il termine "**deployment**" si intende la consegna o rilascio al cliente, con relativa installazione e messa in funzione o esercizio, di una applicazione o di un sistema software tipicamente all'interno di un sistema informatico aziendale. Lo si può di fatto considerare come una fase del ciclo di vita del software, la quale conclude lo **sviluppo** e il relativo **testing** e dà inizio alla **manutenzione**.

Il termine **deployment** è usato a volte al posto del termine **delivery** Utilizzato per descrivere la fase, nell'ambito della realizzazione di un software, di messa in produzione nell'ambiente di utilizzo definitivo.

Principi Della Continuous Integration

- **Mantieni un repository del codice sorgente**
- **Automatizza il build**
- **Rendi il build auto-testante**
- **Tutti eseguono commit alla baseline tutti i giorni**
- **Ogni commit fa partire una build**

- **Fai in modo che il build sia veloce**
- **Eseguire i test in un clone dell'ambiente di produzione**
- **Fai in modo che sia facile prendere le ultime versioni dei pacchetti**
- **Ognuno può vedere i risultati dell'ultimo build**
- **Automatizza i rilasci**

Vantaggi dell'integrazione continua



Maggiore produttività per gli sviluppatori

L'integrazione continua consente al team di migliorare la produttività liberando gli sviluppatori dalle attività manuali e incoraggiando le pratiche che riducono il numero di errori e bug nel software distribuito ai clienti.



Bug individuati e risolti con maggiore prontezza

Aumentando la frequenza del testing, è più facile individuare tempestivamente e risolvere i bug prima che diventino problemi gravi.



Aggiornamenti più rapidi

L'integrazione continua consente di rilasciare aggiornamenti in modo più rapido e con maggiore frequenza.

Continuous integration su Github TRAVIS(CI) & BUDDY

TRAVIS (CI)

Travis CI

- Un'attore fondamentale della CI è proprio il server che esegue l'integrazione, ce ne sono molti, quelli più conosciuti sono sicuramente **Jenkins**, **Hudson CI** ed appunto **Travis CI**.
- Un server CI esegue la build di un progetto Software, quindi di solito, preleva l'ultima versione del progetto da un sistema di **Version Control**, ed esegue uno script che può spingersi fino al **deploy** del software in produzione.

Utilizzare Travis CI con Github

- Come fare:
- Registrarsi a Travis tramite il market-place di github o sulla Homepage del servizio registrandosi con le proprie credenziali Github.
- Fatto questo è possibile tramite il sito stesso scegliere quali repository presenti nel proprio account, vogliamo collegare al servizio Travis.



MarcoEna
5 repositories
Token:

Beta Features

See what's new!

Organizations



TeamEngim
2 repositories

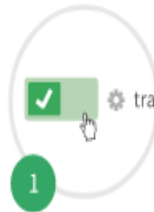
Is an organization missing?

[Review and add your authorized organizations.](#)

MarcoEna

Sync account

We're only showing your public repositories. You can find your private projects on travis-ci.com.



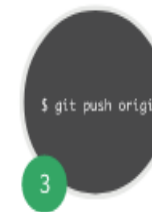
1

Click the repository switch on



2

Add .travis.yml file to your repository



3

Trigger your first build with a git push

Filter repositories



MarcoEna/animated-potato



MarcoEna/Hello-World



MarcoEna/Java-project



MarcoEna/my_first_repository

Integrazione con github

iniziamo l'integrazione inserendo nella root del proprio progetto un file con nome:

travis.yml

All'interno di questo file va specificato il linguaggio con il quale stiamo sviluppando.

language:java

questa è la configurazione automatica. Le convenzioni per quanto riguarda questo caso specifico con linguaggio Java e che il progetto utilizzi una di queste tecnologie:

Oracle JDK 7 (default), Oracle JDK 8, OpenJDK 6, OpenJDK 7, Gradle 2.0, Maven 3.2 and Ant 1.8

in automatico Travis riconoscerà la JDK usata ed eseguirà i test. Tutto questo con una sola riga scritta nel file .travis.yml.

Comunque è sempre buona prassi essere più espliciti possibili specificando la **JDK** e lo script con il quale eseguire i test per determinare se la **build** è corretta o meno.

In questo caso d'esempio infatti avendo un progetto sviluppato con **JDK 1.6** e Maven ho creato un file fatto in questo modo:

language: java

jdk:

- openjdk6

script: cd ../service && mvn test

interessante vedere come tramite parametro script è possibile lanciare dei comandi per testare la **build**. Il processo di test partirà subito dopo aver effettuato una **commit**.

Conclusioni

Quello mostrato è un'esempio molto semplice e limitato per di più legato al mondo Java. **Travis** permette di gestire e testare molte tecnologie, trovate comunque molti esempi e documentazione nel sito di **Travis**.

BUDDY (CI)

Buddy (CI) su Github

- **Buddy** è una continuous integration disponibile sul market place di github, come altre viste in precedenza buddy è utile per testare il codice, e **automatizzare** il processo di **deploy**, che verrà eseguito ad ogni nostra **push** su un branch precedentemente collegato a **buddy**, infatti questo **tool** sviluppa una **pipeline*** sulla quale viene automatizzato il processo di **deploy** e **delivery** della versione di **build**.

*Le pipeline della Continuous Delivery sono il mezzo con il quale le aziende ottengono cicli rapidi di delivery del software.

Vediamo come impostarlo:

Continuous Integration, Deployment & Delivery with Buddy - Mozilla Firefox


Re: what is buddy??? - ma x Continuous Integration, x +

← → ↻ 🏠 ⓘ 🔒 https://buddy.works ... 🔔 ☆ 🔍 Cerca 📄 📁 ☰

Buddy Pricing Enterprise Blog Guides Documentation Sign in [Create free account](#)

Test Environments & Deployment Pipelines

A working website for every branch with deployment automation to production.

 [Sign up with GitHub](#)

Test & Delivery [DEVELOPMENT] [Awaiting push](#)

LAST EXECUTION	EXECUTED REVISION	BRANCH	TRIGGERING
PASSING Executed 7 minutes ago	HEAD	DEV	ON EVERY PUSH

Test & Delivery [PRODUCTION] [Run](#)

LAST EXECUTION	EXECUTED REVISION	BRANCH	TRIGGERING
PASSING Executed 20 hours ago	HEAD	MASTER	MANUAL

my-node-js — **bash**

```
gimbo: ~/Desktop/my-node-js $
```

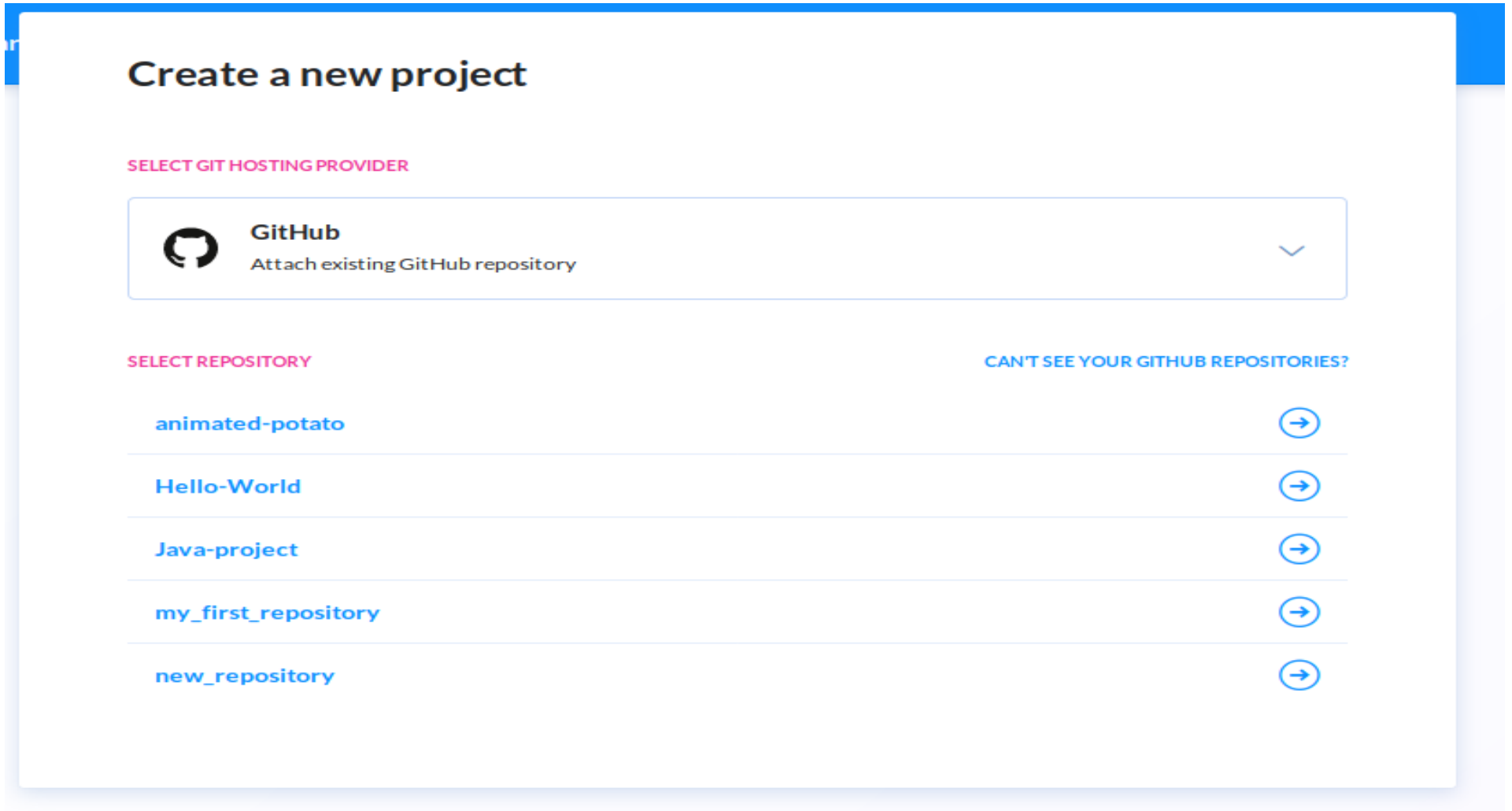
[Add a new pipeline](#)

- Add a new folder
- Clone existing pipeline
- Execute all pipelines
- View mode: Cards
- Submodules: ON
- Environment variables
- YAML configuration: OFF
- Commit commands
- Our IPs

[https://app.buddy.works/register/github?userAgent=Mozilla/5.0+\(X11;+Ubuntu;+Linux+x86_64;+rv:58.0\)+Gecko/20100101+Firefox/58.0&referrer=https://www.google.it/](https://app.buddy.works/register/github?userAgent=Mozilla/5.0+(X11;+Ubuntu;+Linux+x86_64;+rv:58.0)+Gecko/20100101+Firefox/58.0&referrer=https://www.google.it/)

Creiamo un account cliccando su **sign up with github**.


Creiamo un nuovo progetto:




The screenshot shows the 'Create a new project' page on GitHub. At the top, the title 'Create a new project' is displayed. Below it, a section titled 'SELECT GIT HOSTING PROVIDER' contains a single option: 'GitHub' with the GitHub logo and the text 'Attach existing GitHub repository'. A dropdown arrow is visible on the right. Below this, the 'SELECT REPOSITORY' section lists five repositories: 'animated-potato', 'Hello-World', 'Java-project', 'my_first_repository', and 'new_repository'. Each repository name is followed by a blue circular button with a right-pointing arrow. To the right of the repository list, a link 'CAN'T SEE YOUR GITHUB REPOSITORIES?' is visible.


Create a new project


SELECT GIT HOSTING PROVIDER


 **GitHub**
Attach existing GitHub repository


SELECT REPOSITORY

[animated-potato](#) 

[Hello-World](#) 

[Java-project](#) 

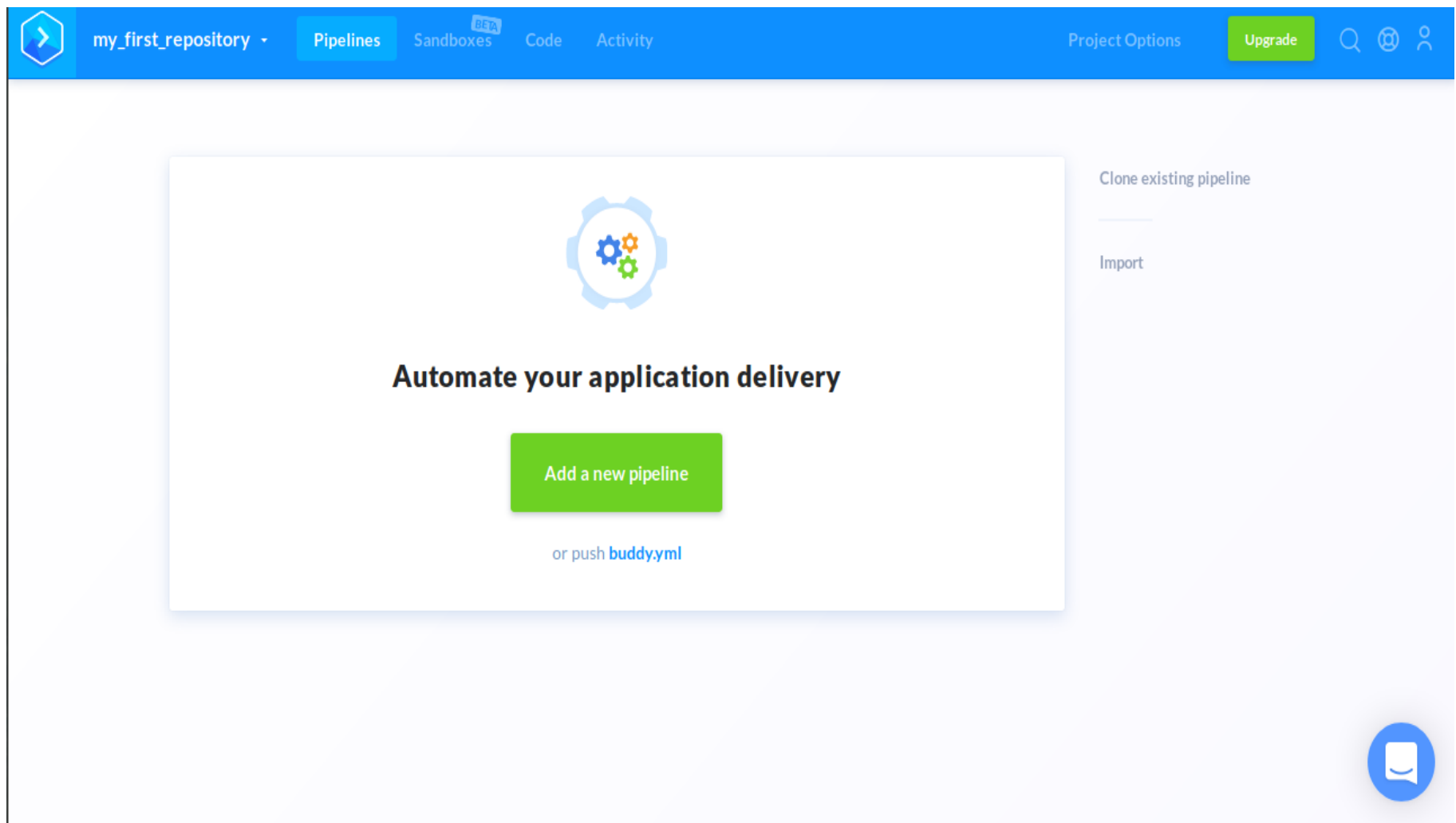
[my_first_repository](#) 

[new_repository](#) 

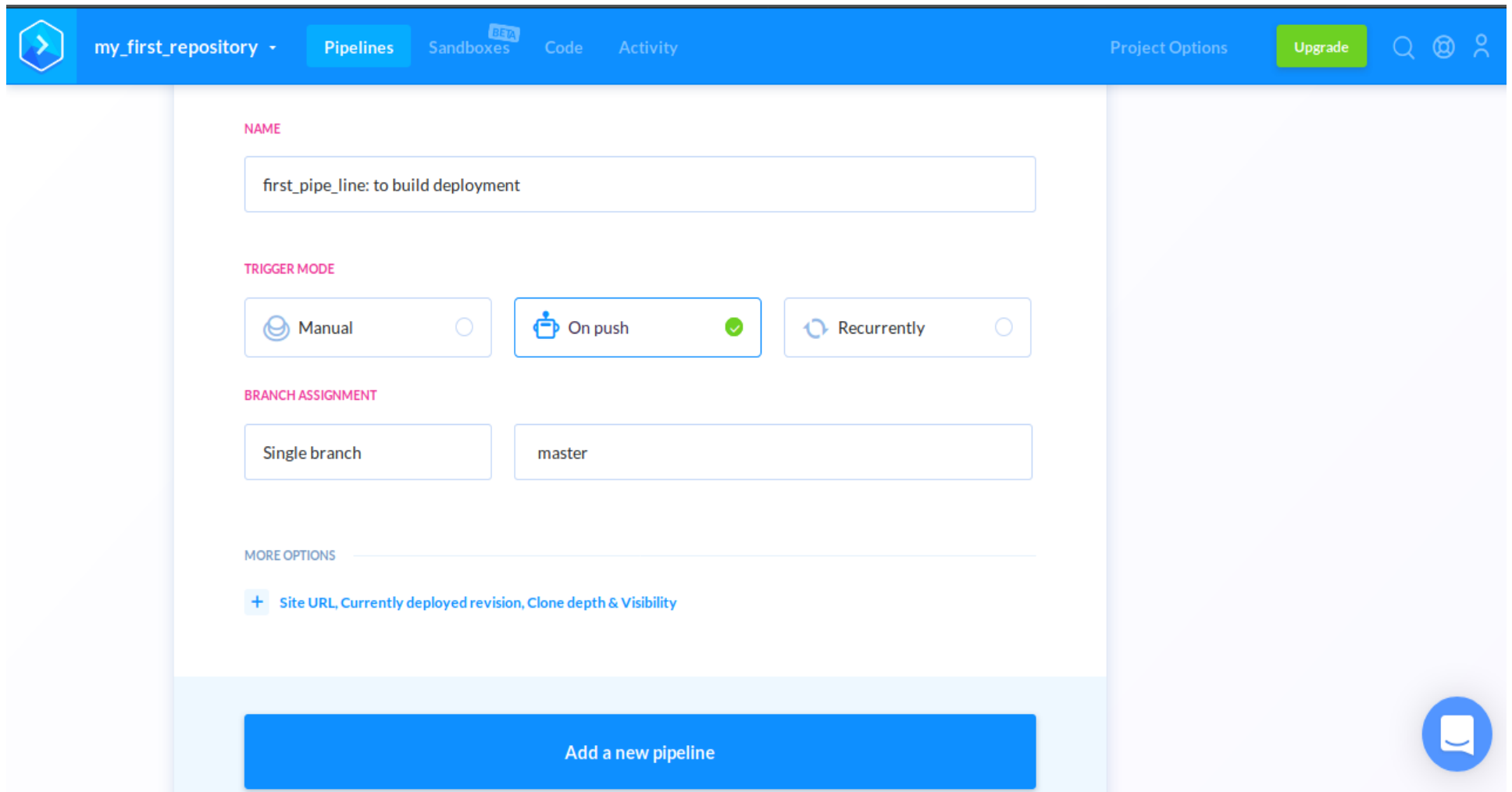
[CAN'T SEE YOUR GITHUB REPOSITORIES?](#)

In automatico ci verranno mostrate le nostre **repository** su **github**. Selezioniamone una e procediamo

Clicchiamo ed aggiungiamo una pipeline:



Diamo un nome alla pipeline, impostiamo la modalità di aggiornamento, che può essere manuale, periodica o come in questo caso dopo ogni push effettuato sul branch selezionato nella casella sottostante.



The screenshot shows the Vercel Pipelines configuration page for a repository named "my_first_repository". The interface is divided into several sections:

- NAME:** A text input field containing "first_pipe_line: to build deployment".
- TRIGGER MODE:** Three radio button options: "Manual" (unselected), "On push" (selected with a green checkmark), and "Recurrently" (unselected).
- BRANCH ASSIGNMENT:** A dropdown menu set to "Single branch" and a text input field containing "master".
- MORE OPTIONS:** A section with a plus icon and the text "Site URL, Currently deployed revision, Clone depth & Visibility".

At the bottom of the configuration area is a large blue button labeled "Add a new pipeline". The top navigation bar includes links for "Pipelines", "Sandboxes", "Code", and "Activity", along with a search icon, a user icon, and an "Upgrade" button.

Qui possiamo selezionare le azioni da aggiungere alla nostra pipeline, essendo numerose è consigliato di valutarle in base al vostro progetto, qui selezionerò quelle di default cliccando su templates of action sets

Pipelines

Add a new action to first_pipe_line: to build deployment

All actions | **Templates of action sets**

Filter actions.. (ex: FTP, Heroku, SSH, build)

SUGGESTIONS

- SFTP
- FTP
- SSH
- Local Shell
- Slack

TRANSFER

Clone from another pipeline

Edit pipeline settings

JUMPTO

- Transfer
- DevOps
- Execute builds & commands in a Docker container
- Deploy to IaaS
- Amazon Web Services
- Google Cloud Platform
- Kubernetes
- Deliver to version control
- Code quality & review
- Performance & app monitoring
- Notifications
- Uptime monitoring

Dovremo poi impostare le credenziali del server sul quale mandare in produzione il **build**. Dopodichè il processo verrà automatizzato con successo.

The screenshot shows the Buddy CI/CD interface. The top navigation bar is blue and contains the following elements: a logo, the text 'my_first_repository', a 'Pipelines' tab, a 'Sandboxes' tab with a 'BETA' badge, 'Code', 'Activity', 'Project Options', an 'Upgrade' button, and search and user icons. The main content area is divided into two sections. The left section, titled 'WHAT TO UPLOAD', contains two options for source upload: 'GitHub repository' (selected with a green checkmark) and 'Pipeline Filesystem'. Below these is a 'Source path' input field with a 'Browse...' button. The right section, titled 'SERVER DETAILS', contains fields for 'Hostname & Port', 'Login', 'Authentication mode', and 'Password'. The 'Hostname & Port' field is split into 'local Host' and '22'. The 'Login' field contains 'Marco Ena'. The 'Authentication mode' field contains 'Password'. The 'Password' field is a text input with masked characters. On the far right, there are links for 'How to use Buddy Params' and 'How to use Environment Variables'. A blue chat bubble icon is visible in the bottom right corner.

my_first_repository - Pipelines Sandboxes ^{BETA} Code Activity Project Options Upgrade

WHAT TO UPLOAD

Source

GitHub repository [?] ☒ Unprocessed source files from the repository

Pipeline Filesystem [?] ☐ Repo clone, processed files & generated artifacts from the actions' shared directory

Source path

/ Browse...

SERVER DETAILS

Hostname & Port Login Authentication mode

local Host 22 Marco Ena Password

Password

.....

How to use Buddy Params

How to use Environment Variables

Ringraziamenti e Contatti

- Ringraziamo tutti per l'interesse dimostrato e vi invitiamo a contattarci per eventuali dubbi o domande sui nostri profili GitHub

-  murty91
-  MarcoEna