Che cos’é GitHub ?

L’idea di rendere il software sempre piú perfetto, ma anche quella di creare delle porzioni di software “riusabili” per piú scopi, sono i concetti su cui GitHub si fonda.

Non é niente altro che un servizio di Hosting su cui lo sviluppatore, dopo essersi registrato, puó caricare il proprio software, rendendolo aperto a chiunque vorrá contribuire allo sviluppo, oppure a chiunque vorrá utilizzarlo nel proprio software.

Un esempio banale, ma molto calzante potrebbe essere: Si vuole creare un software “Macchina”, si avrà bisogno di principi funzionanti come Ruota, Motore, Sedile, Cambio ecc. ecc. In questo caso si avranno le seguenti due scelte:

- si pianifica di voler creare ad uno ad uno tutti i componenti, per poi dedicarsi allo sviluppo della Macchina
- oppure, si può cercare un software libero che include giá i principi funzionanti di Ruota, Motore ecc ecc.

Pensandoci bene, la prima idea potrebbe rivelarsi vincente se, nel proprio progetto, questi concetti assumono caratteristiche uniche e innovative.

La seconda scelta, in caso opposto, diventerà vincente poiché l’investimento di tempo e di energie, é solo per l’idea principale, la Macchina.

In entrambi i precedenti casi lo sviluppo può essere affidato ad un team dove ogni membro (**contributore**) curerà una porzione del sistema.

GitHub, appunto, ci viene in aiuto nel risolvere questi : ci si collega a GitHub, si ricerca il progetto piú adatto e compatibile alle proprie esigenze e che sia scritto con il linguaggio che si vuole utilizzare. In seguito si valuta come distribuire lo sviluppo nel team e quali funzioni nuove integrare o quali delle preesistenti utilizzare.

Anche se banale, questo esempio é quello che permette di spiegare al meglio la questionedi Condivisione, Riusabilitá e di Gestione Delle Versioni

Git e GitHub

Git é un programma per la gestione delle versioni, trovate tutto il necessario sul sito ufficiale o sulla pagina di wikipedia. GitHub, invece, é un servizio di hosting e di team collaboration, basato sulle funzionalità di Git. (differenza sostanziale, da tenere in considerazione)

Sono punti chiave di GIT il ***Repository*** , il ***Branch***, con i rispettivi comandi, ed il ***Workflow*** di sviluppo (di cui ne illustreremo uno ampiamente adoperato).

Il Repository

Sulla base dell'esempio di prima, stiamo per costruire un team di sviluppo per il progetto Macchina. Una volta formato il team, abbiamo bisogno di iniziare a condividere il codice sorgente del nostro progetto. Per questo, il primo step, é la creazione di un contenitore dove andare a salvare il nostro codice, nelle versioni master (ovvero, di produzione), develop (di sviluppo) ed altre. Questo contenitore, appunto, si chiama Repository. (Il nostro Repository master, a sua volta può essere anche un branch o un clone di un progetto esterno open source)
Ogni contributore del progetto, quindi, avrá a disposizione in locale e in remoto, una copia del Repository con tutto il codice sorgente e la documentazione create dagli altri contributori.

Il Branch

Una volta creato il Repository, il team avrá bisogno di iniziare lo sviluppo del software. Prima del promissimo rilascio, ovviamente, il software sará nella cosiddetta fase di sviluppo. Non appena terminata la fase di sviluppo, peró, il team avrá necessitá di "salvare" lo stato del software, magari rendendolo disponibile all'uso. Contemporaneamente, in base alla progettazione del software, alcuni contributori dovranno continuare la fase di sviluppo.

Per non creare caos e stabilire quale elenco di funzionalitá debba contenere una certa versione del software, Git (e GitHub), implementano il concetto di Branch.

Un branch é una sottoparte del Repository che include il codice di una certa versione, derivato direttamente da un altro branch o dalla fusione (merge) di piú branch differenti.

Il Commit, il Push, il Pull e il Merge.

Quando i contributori avranno iniziato a scrivere il codice in modo organizzato, lo faranno in locale, ognuno sul loro PC. Per inviare le proprie modifiche al branch sará necessario eseguire un **Commit**. Possiamo definire, quindi, il commit come l'azione con cui si decide di trasferire il cambiamento apportato al codice, rendendolo disponibile a chiunque abbia accesso al Repository.

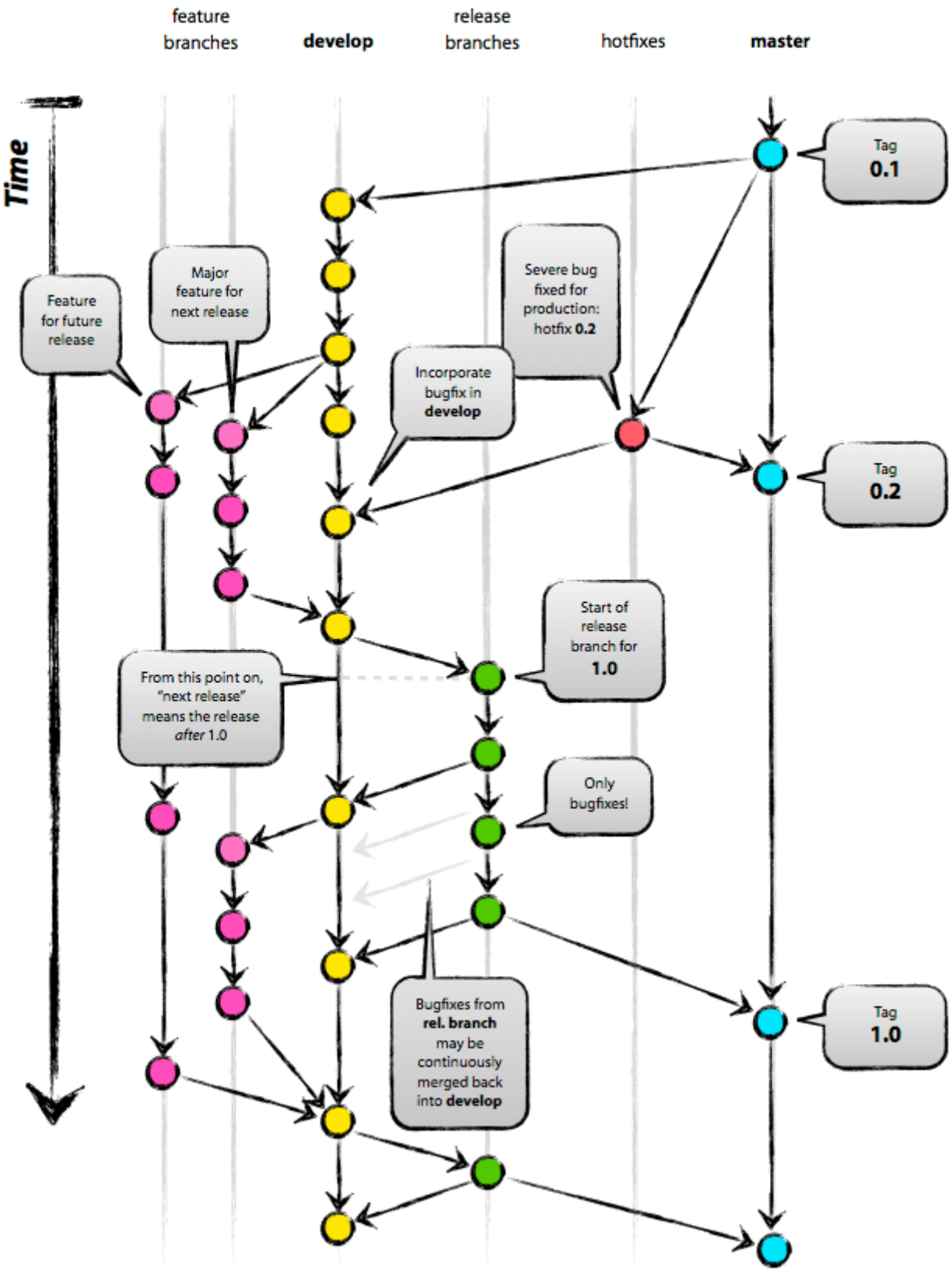
Con l'azione di commit, in veritá, non si invia nulla al repository remoto, e quindi agli altri contributori, ma si genera solo un pacchetto contenente le modifiche. Per inviare il pacchetto bisogna usare il comando **Push**, mediante il quale il Repository acquisisce le modifiche da un contributore, rendendole disponibili agli altri.

Ogni contributore, per acquisire nel proprio Repository locale le modifiche inviate dagli altri, deve effettuare il comando di **Pull**, mediante il quale il proprio Repository locale viene allineato con le modifiche presenti sul remoto.

L'operazione di allineamento, in alcuni casi potrebbe richiedere l'elaborazione del "**merge**" manuale, soprattutto quando contributori diversi lavorano su un medesimo file.

Il workflow piú usato

Il branch iniziale, presente in ogni Repository, é il master. A partire dal master, in genere, si genera il branch di sviluppo "develop". Detto questo, la regola sull'uso dei branch non é unica, ma una delle soluzioni piú diffuse é quella riportata in nell'immagine seguente:



Come analizzare questo grafo? Semplice, si legge da destra a sinistra e dall'alto verso il basso. L'asse verticale é il tempo, mentre le linee sono i branch. I punti sulle linee sono i commit.
Il team concorda la prima versione del progetto nel branch master (o main), taggandone la versione come v 0.1. A partire da questa versione, si crea il

branch develop, su cui alcuni contributori iniziano a lavorare e inviando i **commit** al repository (**punti gialli** sulla linea develop). I branch piú a sinistra sono denominati feature-branch, ovvero dei branch che racchiudono una funzionalitá più complessa, che va gestita a parte rispetto alla normale fase di sviluppo.
Quando il grafo si sposta da destra a sinistra si sta derivando, ovvero si crea un nuovo branch a partire da uno giá esistente, ad una precisa versione (o hash). Tutto il codice del branch a destra sará quindi incluso in quello di sinistra.

Quando il grafo si sposta da sinistra a destra, si sta compiendo un merge, ovvero si stanno fondendo le modifiche effettuate su un branch, ad esempio di features, sul branch di develop. Il branch risultante avrá le modifiche effettuate mediante i commit sul branch stesso, piú i commit effettuati sul branch che si sta fondendo. E' buona regola cancellare tutti i branch che sono reputati "binari morti" per non avere confusione nel Repository.

I merge, quindi, sono effettuati mano a mano da sinistra verso destra, fino ad arrivare al master. La politica scelta per la gestione dei rilasci, é da definire come meglio si crede.

Per sperimentare quali siano i concetti Git finora esposti utilizzando GitHub sono, qui di seguito illustrate, le seguenti azioni, ossia:

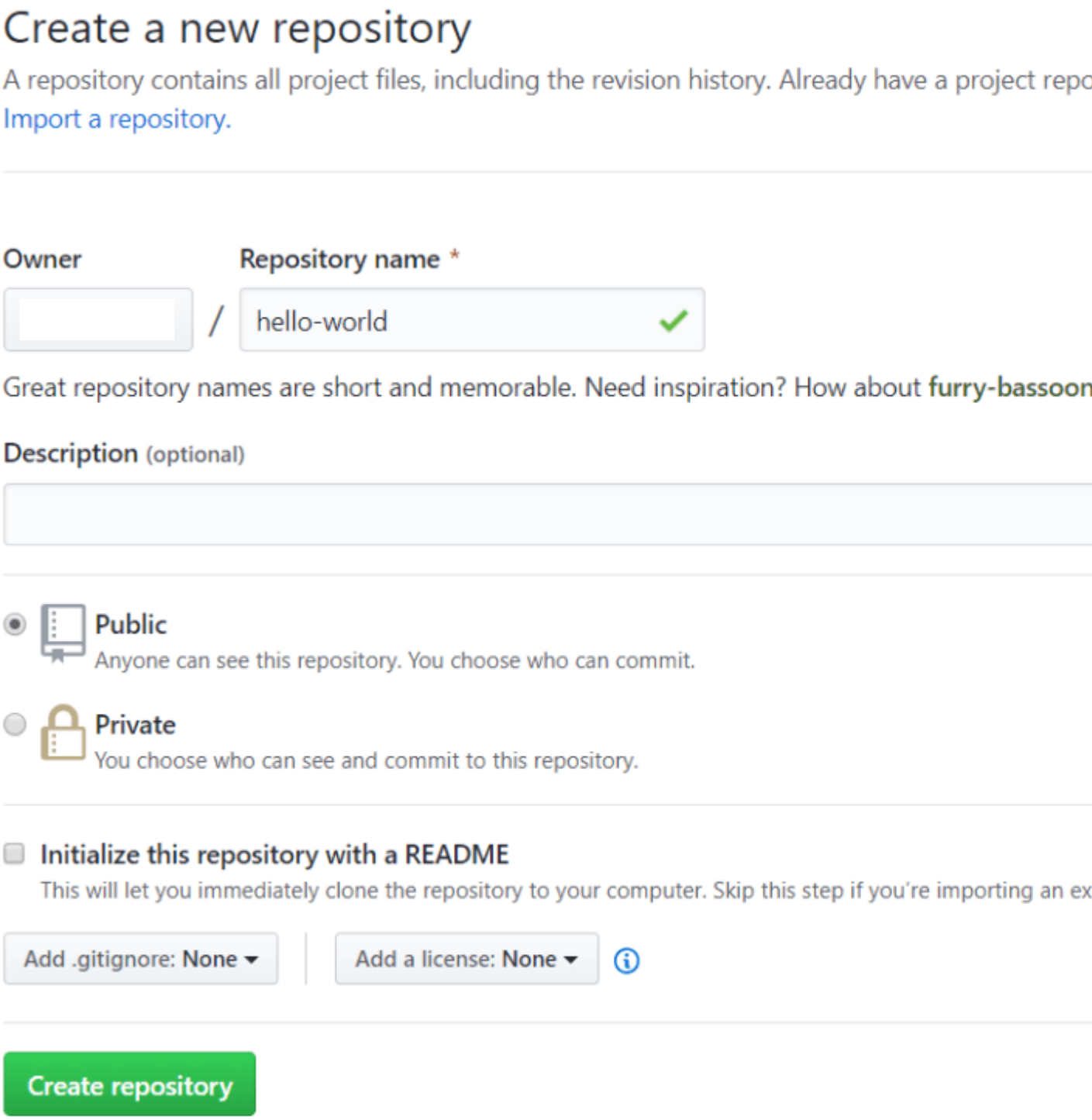
- Creare un nuovo Repository GitHub
- Creare un nuovo Branch develop
- Apportare modifiche ed inviarle mediante Commit
- Aprire una Pull Request
- Effettuare il Merge

Per proseguire, è necessario creare un proprio account sul sito github.com

Creare un nuovo Repository

Un repository, in genere, é pensato per ospitare un solo progetto. In genere contiene il codice sorgente organizzato per cartelle e puó contenere anche tutti gli assets (immagini, CSS, ecc) utilizzati nel progetto.

Nella parte destra del sito, si trova l'icona "+", cliccandoci si potrà scegliere "new repository".



Nella schermata di creazione si potrà scegliere:

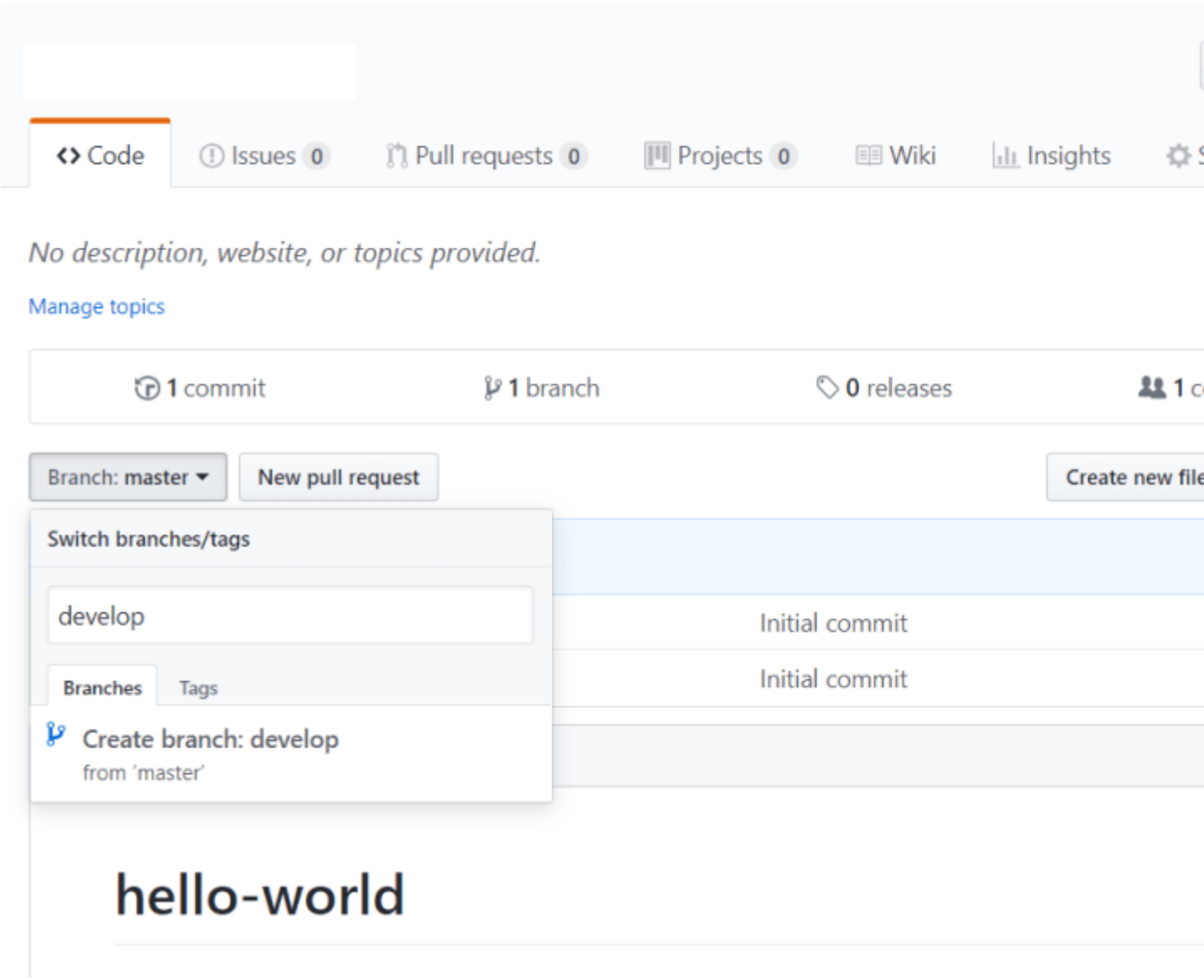
- Il nome del Repository
- Una descrizione
- Se Renderlo Pubblico o Privato

Molto utile é lo strumento di aggiunta del .gitignore (file che permette di escludere path o files dal repository) e il file di licenza, se si sa quale usare.

E’buona regola corredare il repository di un file README, che contenga la descrizione del progetto. Procedere dunque alla creazione del Repository...

Creare un nuovo Branch

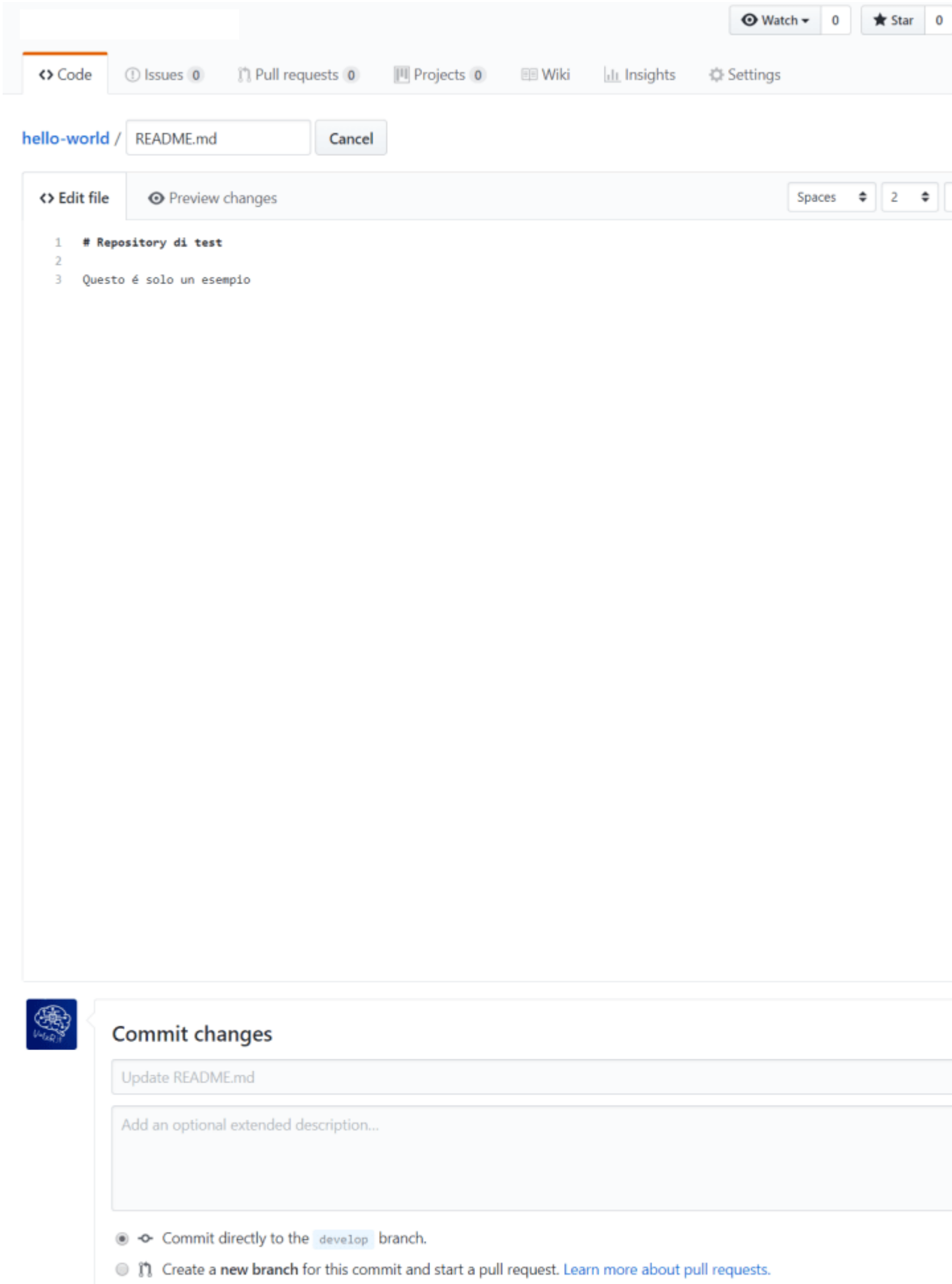
Procediamo alla creazione del branch develop per rendere possibile l’adozione del workflow sopra descritto: dal dropdown menu branchs, si potrà scriviere il nome del nuovo branch e cliccare su “create branch: develop”.



Una volta creato il branch develop, si potrà notare che includerá tutto il codice del branch da cui é nato, ovvero master.

Apportare le modifiche ed inviarle al Repository

A questo punto il nostro Repository avrá due branch separati, uno chiamato master, l’altro develop. Passando al branch develop mediante il dropdown menu branches, andiamo ad editare il file README, aggiungendoci dei testi.


Watch 0
Star 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
hello-world / README.md
Cancel
Edit file
Preview changes
Spaces
2
1 # Repository di test
2
3 Questo é solo un esempio
Commit changes
Update README.md
Add an optional extended description...
Commit directly to the develop branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.

Notare come, in basso all’editor, si trovi il form in inserire, il titolo del commit e una descrizione. Queste informazioni servono da guida agli altri contributori, che potranno identificare la natura del vostro nuovo codice. Nell’esempio, scriveremo: Modificato il file README.

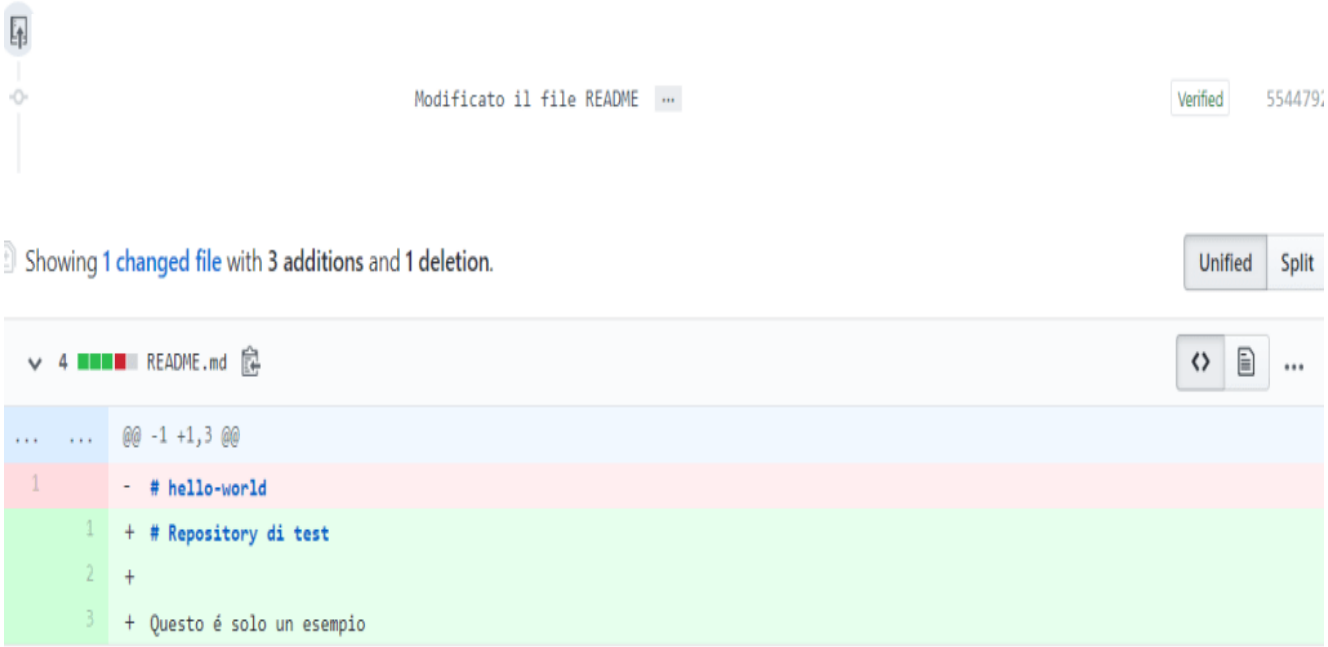
A commit inviato, il branch develop mostrerá le tue ultime modifiche.

Notare che questo modus operandi é totalmente esemplificativo. A seconda del linguaggio di programmazione scelto, si farà uso di un IDE che avrá sicuramente il supporto a GIT, ad esempio Visual Studio Code. Tutto il flusso di modifica, push e pull, sará gestito dalla IDE stessa

Aprire una Pull Request

Come farà un contributore a far capire che le sue modifiche sono pronte per essere spalmate sul branch master ? Egli procederà con la creazione di una Pull Request come di seguito indicato:

Dal menu Pull Requests del Repository, cliccherà sul pulsante New Pull Request. Inizierà una procedura guidata per il merge delle proprie modifiche.



Una volta confermato il merge, scegliendo un commento al merge, potrà terminare la procedura e chiuderlo. Da questo momento in poi, il branch master avrá lo stesso codice del branch develop.