GitHub è un servizio pubblico di gestione del codice sorgente basato sul sistema Git, inventato da Linus Torvalds anima e per anni programmatore di Linux.

I sistemi di controllo del codice sorgente sono una vecchia conoscenza del mondo della programmazione. Dato che spesso per risolvere taluni problemi si fanno esperimenti che convivono col codice in produzione, che va manutenuto, i sistemi di controllo del codice sorgente (SCCS) offrono vari servizi per effettuare queste operazioni. Principalmente la gestione dei rami di sviluppo e la storicizzazione delle versioni. In pratica con un sistema di questo tipo ciascun programmatore è in grado di provare modifiche senza correre il rischio che queste inquinino il codice testato, in correzione e in produzione ma anche di gestire le versioni, cioè di ricostruire esattamente il contenuto di tutti i file sorgenti che in un dato giorno erano stati inseriti nel sistema.

A differenza di altri sistemi, GIT offre la possibilità di gestire sia un sistema locale privato sia un sistema globale condiviso.

Non entro nei dettagli di GIT, che peraltro sono molto complessi, ma GitHub è un servizio globale che fornisce un server che copre la parte Git condivisa.

Il servizio è complesso e vi sono varie forme di uso, ma per la programmazione open source il servizio è gratuito, quindi è possibile crare una propria pagina (la mia è questa fjovine (Francesco Iovine)) in cui memorizzare propri contributi liberi e open source che derivano o da sviluppi completamente indipendenti (io ho solo progetti originati da me e contenenti solo codice mio) o da fork, cioè che partono da un progetto già esistente e supportato magari su GitHub e lo estendono in altre direzioni, senza ovviamente contaminare il sorgente da cui partono.

Dunque questo è un primo uso per uno sviluppatore: una vetrina che permetta a eventuali datori di lavoro/committenti di misurare direttamente le tue competenze guardando il tuo modo di scrivere codice dai sorgenti.

Fresco di acquisizione da parte di Microsoft, GitHub è il cuore di un servizio di hosting di repository Git, ovvero una gestione del codice sorgente basata su cloud… ma questo è solo l’inizio. GitHub implementa infatti anche funzionalità per la revisione del codice (richieste di pull, diff e richieste di revisione), gestione del progetto (incluso monitoraggio e assegnazione dei problemi), integrazioni con altri strumenti di sviluppo, gestione del team, documentazione e “social coding”.

Molto simile a una sorta di social network per sviluppatori, GitHub è un ambiente aperto in cui i programmatori possono condividere e collaborare liberamente (anche ad hoc) su codice open source. GitHub rende facile trovare codice utile, copiare repository per uso personale e inviare modifiche ai progetti di altri, diventando sede di praticamente ogni progetto open source di qualsiasi importanza.

Controllo della versione Git

Prima di poter capire cosa fa GitHub e come funziona, dobbiamo capire cos’è Git. Git è un sistema di controllo della versione distribuito originariamente scritto da Linus Torvalds nel 2005 per e con l’aiuto della comunità del kernel di Linux. Git è nato inizialmente come uno strumento da riga di comando, adattandosi alla sua origine nella comunità del kernel Linux. Potete ancora usare la riga di comando di Git se volete, ma non è più obbligatorio.

Al posto (o in aggiunta) della riga di comando, è infatti possibile utilizzare il client GitHub gratuito per Windows o Mac o qualsiasi altra GUI per Git o un editor di codice che si integri con Git. Tutte queste opzioni sono inizialmente più facili da usare rispetto alla riga di comando. La riga di comando Git è preinstallata sulla maggior parte dei sistemi Mac e Linux e supporta tutte le operazioni; le GUI in genere supportano un sottoinsieme di operazioni Git frequentemente utilizzato.

Git è diverso dai sistemi di controllo delle versioni precedenti come Subversion in quanto è distribuito piuttosto che centralizzato. È anche abbastanza veloce, soprattutto dal momento che la maggior parte delle operazioni avviene sul vostro repository locale. Tuttavia, l’utilizzo di Git aggiunge un livello di complessità. L’invio di codice al repository locale e l’invio di commit a un repository remoto sono passaggi separati. Quando i team dimenticano questo (o non gli è stato insegnato), si può arrivare a situazioni in cui diversi sviluppatori lavorano con basi di codice divergenti.

Un repository Git remoto può trovarsi su un server o su un altro computer dello sviluppatore e Ciò consente molti possibili flussi di lavoro per i team. Un flusso di lavoro comune prevede l’utilizzo di un repository server come repository “blessed”, al quale viene inviato solo codice verificato e ben testato, spesso tramite una richiesta pull emessa dal repository di uno sviluppatore.

Articolo VBOX

Il termine virtualizzazione fa riferimento alla possibilità di astrarre le componenti hardware del PC o di un qualunque dispositivo in modo da renderle disponibili sotto forma di risorse virtuali.

Su una macchina ospitante (detta host), indipendentemente dal sistema operativo installato, è possibile caricare ed eseguire più sistemi operativi, anche molto diversi l'uno dall'altro.

Se sulla macchina "host" fosse installata, ad esempio, una certa versione di Windows, all'interno della macchina virtuale, grazie alla virtualizzazione, si possono installare ed eseguire altre versioni di Windows, una distribuzione Linux oppure altri sistemi operativi.

Alla macchina virtuale, all'interno della quale si installerà il sistema operativo, vengono destinate parte delle risorse fisiche della macchina "host" in uso.

Eventuali contese per l'utilizzo di una stessa risorsa, vengono gestite dai software che si occupano della virtualizzazione (Virtualbox è uno di essi) e in particolare dal componente detto hypervisor.

Sebbene le macchine virtuali facciano già tanto, le sole non basterebbero a rendere il nostro viaggio liscio come la seta. Uno degli scenari più noiosi, infatti, è proprio la preparazione dell’ambiente di una macchina virtuale: dobbiamo avviare la macchina, entrarci, fare tutti i nostri setup, rigenerare un pacchetto di diversi gigabyte e condividerlo con il gruppo.

Se un giorno ci rendessimo conto di dover fare un upgrade di una componente: stesso giro e stesso scambio di gigabyte. Decisamente poco pratico…

Inoltre, la cosa peggiore, è che tutto questo processo di configurazione della macchina virtuale, anche noto come provisioning, non è tracciabile (a meno di qualche passaggio manuale) e quindi non facilmente ripetibile.

Vagrant invece, consente di avere delle macchine virtuali completamente “scriptabili” sia in fase di configurazione (RAM, networking, spazio disco…) che di provisioning (installazione di MongoDB, RabbitMQ…). Ognuno di questi script potrà poi essere aggiunto al nostro source control preferito e arrivare a versionare i vari scenari realizzando quella pratica nota come Infrastructure as code.

I vantaggi di questo approccio sono evidenti visto che non avremo più necessità di dover trovare noi un modo per condividere le virtual machine. Per comprendere la semplicità di utilizzo, basti pensare che una macchina virtuale Linux è possibile avviarla con due semplici comandi inseriti in un prompt di Windows.

PuTTY, un software open source che ora come ora è utilizzato per lo più dai web master in merito alla gestione (in remoto) dei propri server, usando SSH. Il software, per Windows, rende possibile l’interazione con i sistemi Unix remoti grazie all’utilizzo dei protocolli SSH, Rlogin e Telnet.

Il suo funzionamento, tutto sommato, non è poi così difficile come sembra al primo approccio. Per iniziare ad usarlo, si fa partire il programma attraverso Microsoft Windows, di poi è importante impostare tutti i parametri base necessari per la connessione al server remoto (ad esempio l’IP, l’user name e la password).

In informatica, con il termine PuTTY, ci si riferisce ad un client (il cui sviluppo risale ai primi mesi del 1999 da parte del programmatore Simon Tatham) con i protocolli SSH, Telnet, rlogin e TCP che permette di stabilire una sessione remota cifrata ad un altro computer emulando un terminale, effettuando il login con l’IP del computer a cui ci si vuole connettere.

Ant è una libreria JAVA sviluppata dalla Apache che permette di automatizzare il processo di sviluppo di applicazioni Java. Con Ant, infatti, è possibile creare un progetto che compila, genera la documentazione, realizza file jar, war ed ear ed effettua il deploy di un’applicazione web su un application server, tutto con il semplice lancio di un comando.

Ant è un programma che si lancia da riga di comando. I comandi che Ant esegue sono letti da un file XML, di solito chiamato build.xml. In questo file bisogna definire le operazioni (target) disponibili e, per ciascuna di esse, i comandi da eseguire (task).

Il file build.xml è composto da un tag <project> che contiene a sua volta tutti i comandi disponibili, denominati target.

Ciascun target può avere target dipendenti, ciò vuol dire che, se viene richiesta l’esecuzione di un target, Ant eseguirà prima i suoi target dipendenti.

Analizziamo i comandi principali (task) che Ant mette a disposizione:

<echo>: permette di scrivere un messaggio sulla console; esempio: <echo message="Questo è in nostro primo progetto Ant"/>;

<property>: permette di definire una o più property. È possibile anche definire le property in un file esterno. Le Properties sono case sensitive ed immutabili: chiunque setti per primo il valore per la property la congela per la restante esecuzione dello script. Le property saranno poi visibili all’interno del file build.xml utilizzando la seguente sintassi: ${nomeVar};

<jar>: permette di creare un file jar. È necessario definire il nome del file jar da creare e la directory contenente i file class. È possibile definire eventualmente sia necessario anche le directory o i file da escludere;

<war>: permette di creare un file war. È possibile definire i file che andranno nella root (fileset), le librerie e i file classes che andranno nella directory WEB-INF.

<javac>: permette di compilare i file java. Ant scandisce ricorsivamente la directory specificata nel parametro srcdir alla ricerca di file java per i quali il relativo file .class non è presente o è meno recente del sorgente. Con questo controllo viene evitato di compilare i file già compilati che non hanno subito modifiche rispetto all’ultima compilazione.

Pagina: 12