# Jenkins: Server open source per Continuous Integration and Continuous Delivery

## INSTALLAZIONE:

* L’installazione di Jenkins si trova in una macchina virtuale che risponde all’indirizzo: 192.168.2.43.

Il nome della macchina è: Jenkins-DEB

* La macchina a 64 bit
* Il Sistema operativo utilizzato: Linux Debian Jessie 8.5
* Gli Utenti per accesso alla macchina:
  + Username: root Password: Passw0rd!?
  + Username: user Password: qwertyuiop
  + Username: sshusr Password: poiuytrewq

Entrambi gli utenti sono sudoers. Root è abilitato all’ssh

* Per accedere alla macchina da remoto occorre eseguire l’Installazione di Putty in locale per accesso ssh
* Nella macchina che ospita Jenkins è necessario installare tutti i componenti necessari alla compilazione del codice sorgente e dunque riprodurre l’ambiente di sviluppo. Nel nostro caso:
  + Git
  + Maven
  + HSQL
* Prima di installare Jenkins, per avere un ambiente Java identico a quello di sviluppo, il team ha ritenuto opportuno disinstallare la versione OpenJDK di default presente su Debian a favore della verione JDK 1.8 di Oracle. Qui di seguito il link che spiega i passi da seguire: <https://wiki.debian.org/JavaPackage>

In sostanza:

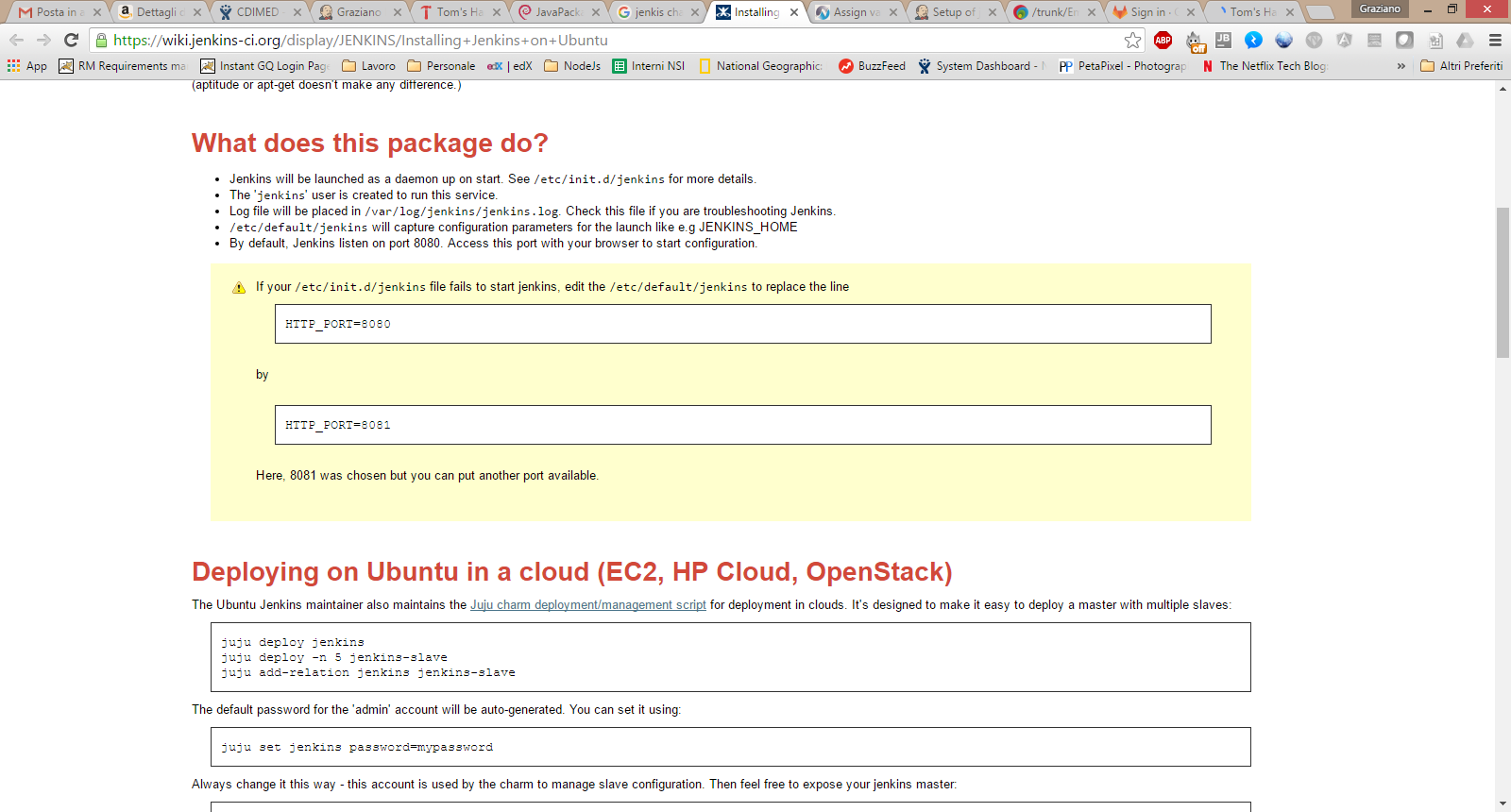
1) Download dal sito sun della versione di java per Linux a 64 bit

2) Trasformazione del pacchetto tar in .deb

3) Installazione del pacchetto: Sudo dpkg –i nome pacchetto.deb

4) Settaggio della java 1.8 come sdk di default

* Download del pacchetto Jenkins con il comando: apt-get install jenkins
  + Jenkins viene installato come servizio (lanciato come demone) attraverso uno script di startup in */etc/init.d/jenkins*  eseguito da un utente di sistema chiamato “jenkins” definito sempre all’interno dello script
* Di default il War di Jenkins è nel percorso: *usr/share/jenkins*
* La home directory in*/var/lib/jenkins*
* In generale per lanciare Jenkins è necessario:
* $ **sudo /etc/init.d/jenkins start**
* La porta di default è la 8080 (<http://localhost:8080/>).
* Per stoppare il servizio:
* $ **sudo /etc/inid.d/jenkins stop**
* Jenkins scrive i log files in */var/log/jenkins/jenkins.log*.
* I parametri di configurazione sono in /etc/default/jenkins  e.g JENKINS\_HOME
* <https://wiki.jenkins-ci.org/display/JENKINS/Starting+and+Accessing+Jenkins>
* Per modificare la porta di default:



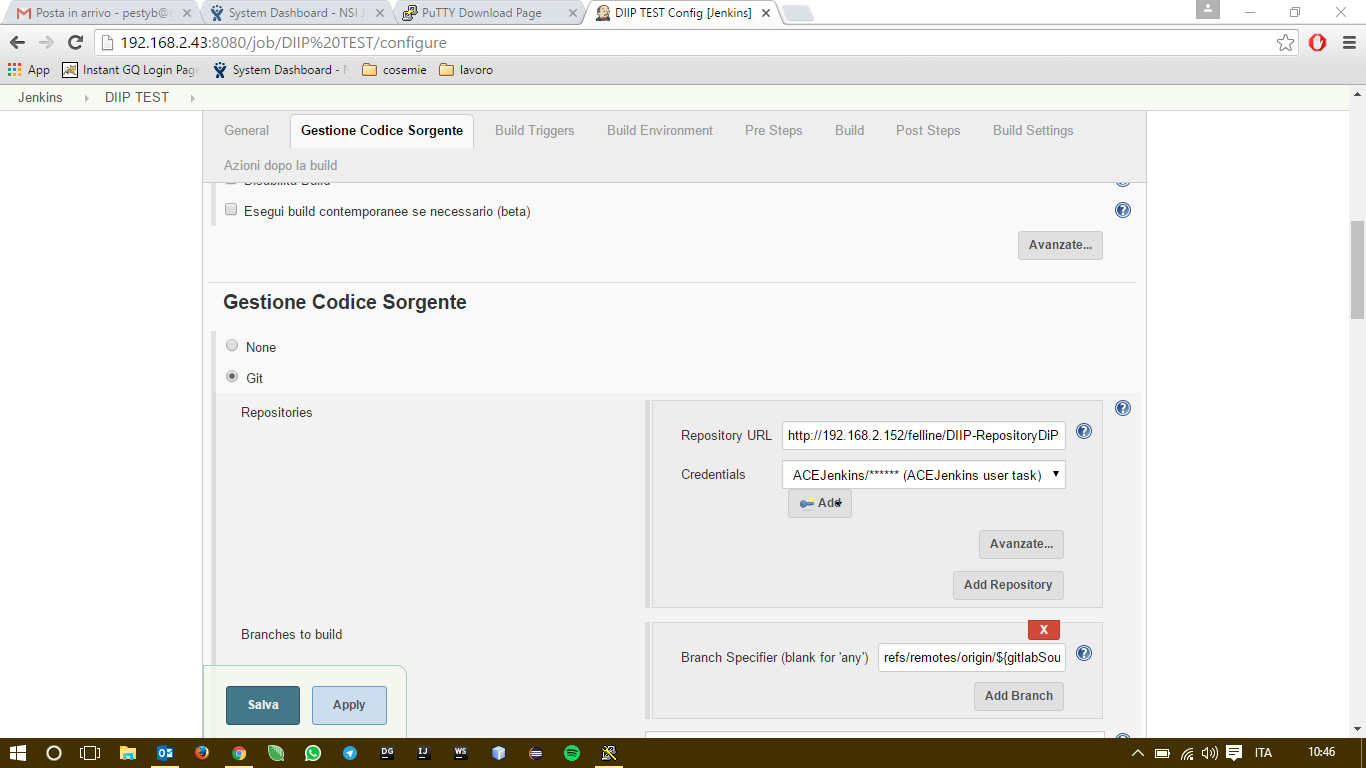
* Il server web che ospita Jenkins è Jetty ed è bundle nel war.

ACCESSO A JENKINS-

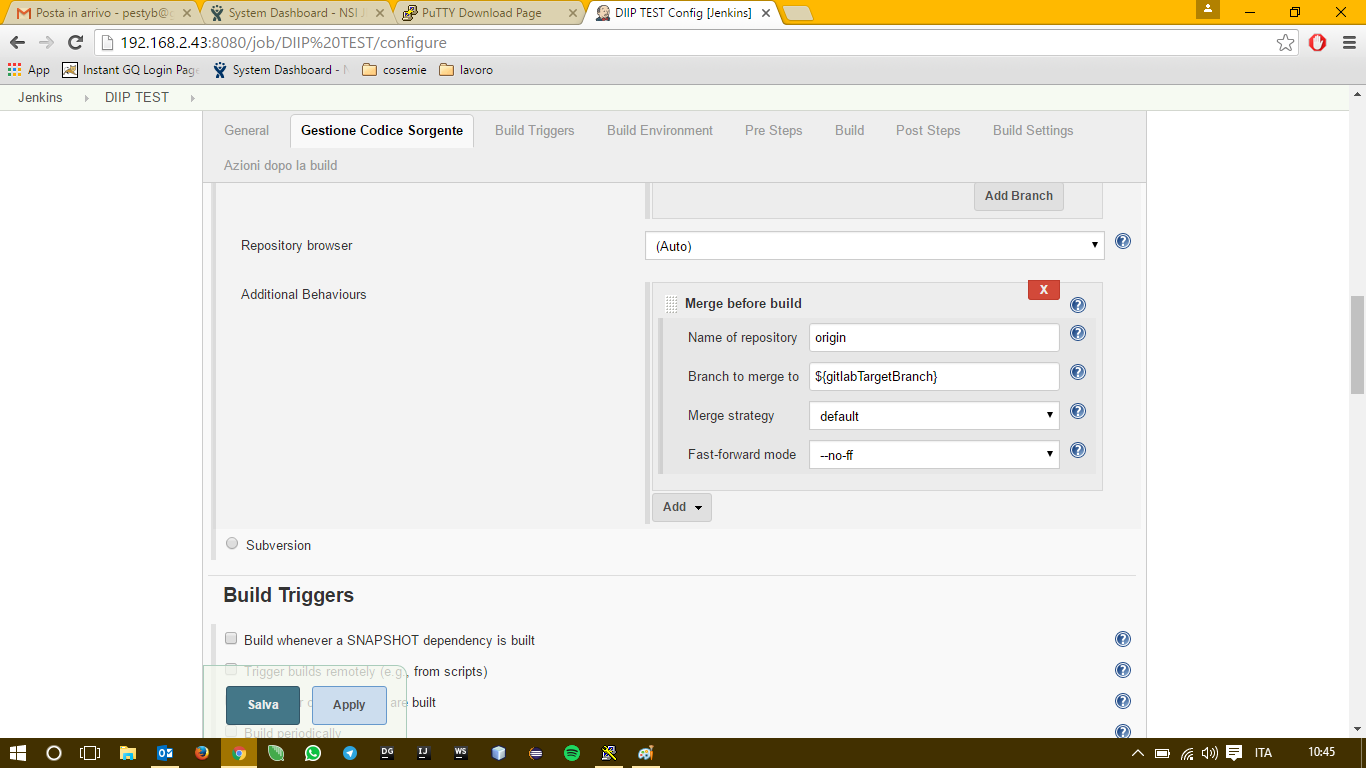
* L’interfaccia di configurazione di Jenkins è raggiungibile da: <http://192.168.2.43:8080>
* Al primo accesso viene richiesta una password recuperabile solo se si è amministratori della macchina ed è raggiungibile ad un percorso specifico
* Installazione dei plugin: Select plugin to install e abbiamo selezionato:
  + JUnit
  + gitserver
  + Maven Integration Plugin
* Creazione del primo utente amministrativo:
  + Aceteam
  + 12345678
* Configurazione da Menù Configura Jenkins:
  + Configura Sistema:
    - Executors:1
  + Configurazione gitserver
    - Accesso a gitserver
    - Creazione di un utente per la gestione del repository: [aceteam@nsi.it](mailto:aceteam@nsi.it); pwd: 12345678
  + Creazione del repository
  + Creazione utenti e ruoli
* Ritorno su Jenkins per Impostazione connection con gitserver
  + Connection name
  + host URL: assicurarsi di non utilizzare il dns se non cofigurato nella macchina che ospita Jenkins. Nel nostro caso abbiamo utilizzato l’indirzzo ip di gitserver 192.168.2.152
  + Credential: utilizzare il token privato (Account in gitserver) generato in gitserver per impostare le credenziali di accesso da parte di Jenkins a gitserver. In alternativa si potrebbe utilizzare l’autenticazione classica via username e password.
  + Test Connection
* Configurazione Creazione Nuovo Job:
  + Nome: Task per i test
  + Selezionare il plugin Maven come tipologia del task
  + Gestione Codice Sorgente:
    - Repository Url: copiare l’url del repository da gitserver.

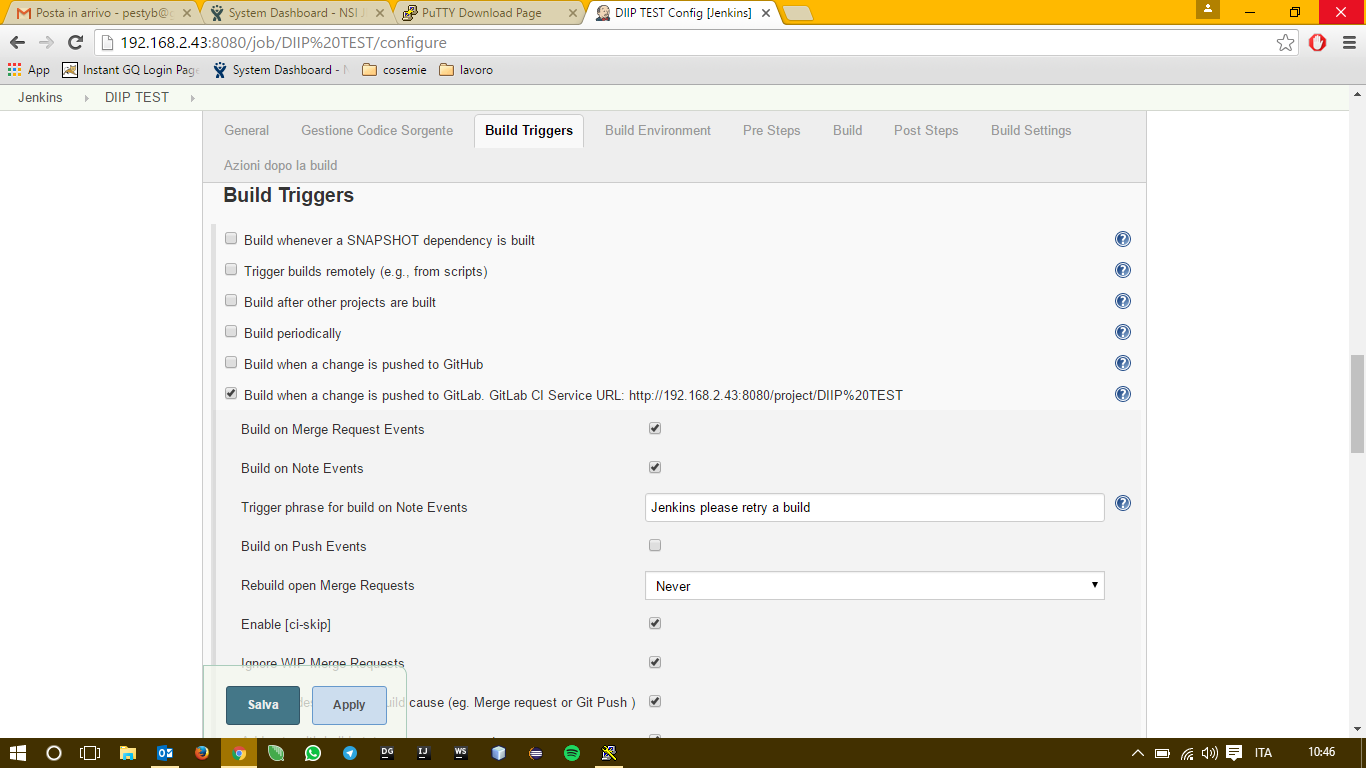
Es: <http://gitserver/acerioni/rimborsi_spese.git>

* + Reimpostare le credenziali di accesso a gitserver:
    - Api token
    - ID
  + Build Triggers:
    - Build when a change is pushed to gitserver
    - Nel nostro caso solo se viene scatenato un evento di merge request su gitserver
* In gitserver -> Repository -> Setting -> Web Hooks
  + La creazione di un Web Hooks consiste nell’indicare in gitserver l’Url Jenkins da notificare quando accade un evento.
  + Nel nostro caso l’evento da notificare a Jenkins è l’evento di Merge e sfleggare l’autenticazione SSL se Jenkins non è in https.
* Su Jenkins:
  + Gestione Codice Sorgente: dobbiamo indicare a Jenkins quali parametri della post inviata da GitLab leggere per recuperare il codice sorgente nel repository Gitlab con il quale effettuare il merge in Memoria (vedi spiegazione sotto Merge Before Build) ed eseguire uno o più goal maven.
  + Configurare la sezione Additional Behavior:
    - Branches to build: è il branch etichetatto come source in Gitlab

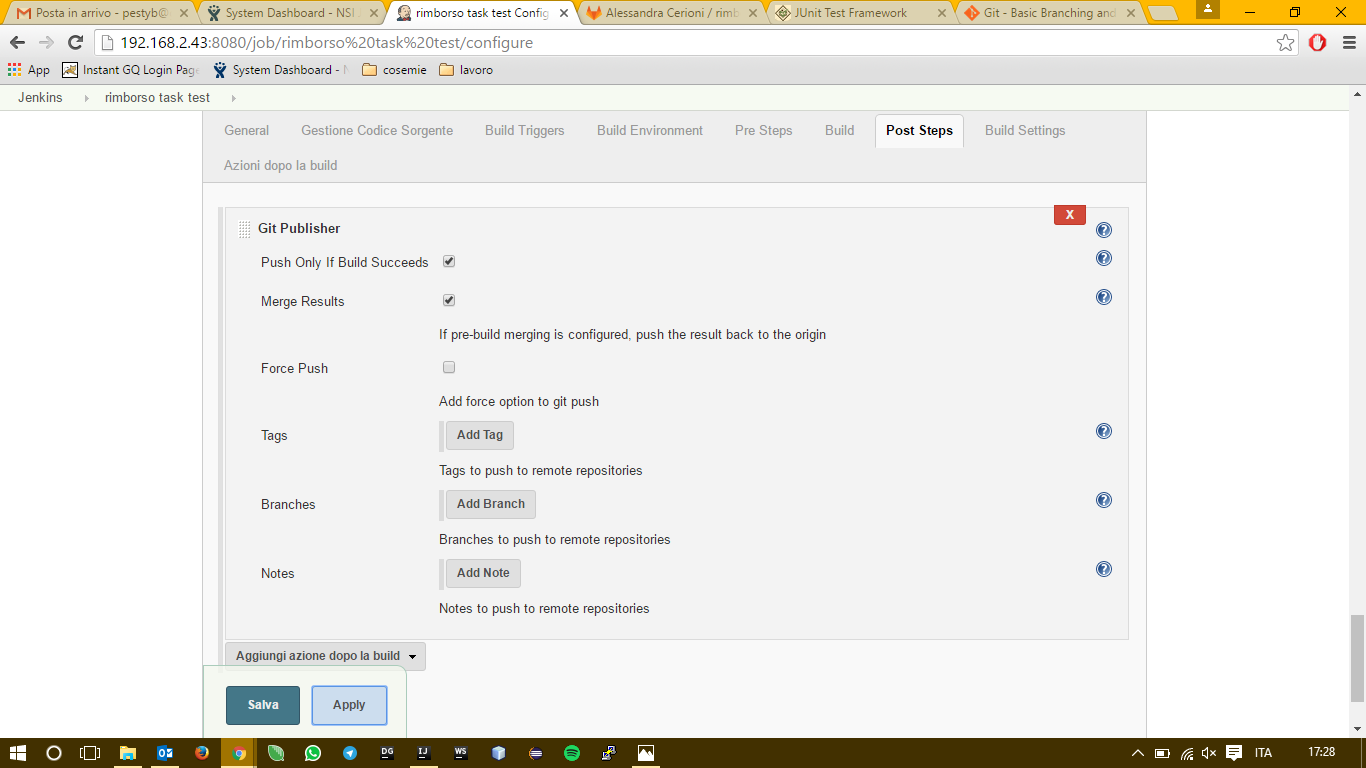


* + - Il target Branch è il branch principale (es. master o develop)
    - Add Merge Before Buid: Jenkins utilizza un workspace dedicato per ogni singolo task dove clona il codice sorgente attraverso il comando git clone:



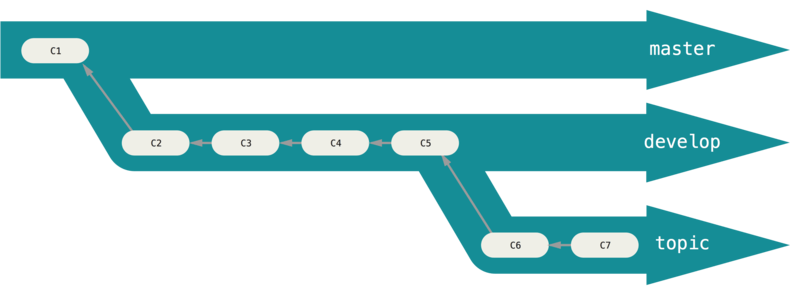


* Al termine del merge nel suo worskpace, Jenkins se la build avviene con successo ovvero i goal stabiliti sono stati superati, fa push su GitLab della merge fatta in fase di prebuild



## Struttura progetto su GitLab

* Nel repository GitLab, abbiamo stabilito di utilizzare la seguente struttura suddivisa in branch:
  + Master (branch della versione di produzione)
  + Develop (branch della versione di sviluppo e test)
  + Topic (branch di ogni sviluppatore)



## Creazione del progetto su GitLab

La creazione del progetto su GitLab avviene per caricamento del progetto a partire da una prima copia locale e seguendo i seguenti passi:

* Creazione di una cartella in locale (nel pc dello sviluppatore) contenente la prima versione del progetto