

# Integració Contínua amb Jenkins



## Introducció

*Jenkins* és un servidor d'automatització amb codi obert programat en Java. *Jenkins* és una eina que ajuda a automatitzar la part no humana del procés de desenvolupament del software, amb una integració continua facilitant així els aspectes tècnics de lliurament continu.

The screenshot shows the Jenkins web interface. At the top, there's a header with the Jenkins logo, a search bar, and the user 'Claudia Meindl' with a 'log out' link. Below the header, the breadcrumb trail is 'Jenkins > Rollout > Rollout-system > #36'. On the left sidebar, there are links: 'Back to Project', 'Status', 'Changes', 'Console Output' (highlighted), 'View as plain text', 'View Build Information', 'Git Build Data', and 'Previous Build'. The main content area is titled 'Console Output' and displays the following log text:

```
Skipping 1.196 KB.. Full Log
KoEttlJQNjBwdnF2sVayAwD4vuoDpQAAAA==[0mskipping: [teamwiesn.com]

TASK [smtp-server : include] *****
skipping: [teamwiesn.com]

TASK [ssl : include] *****
included: /var/lib/jenkins/jobs/Rollout-system/workspace/roles/ssl/tasks/setup.yml for
teamwiesn.com

TASK [ssl : install ssl packages] *****
ok: [teamwiesn.com] => (item=[u'ssl-cert', u'sslscan'])

TASK [ssl : Install ssl certificates] *****
ok: [teamwiesn.com] => (item=teamwiesn.crt)

TASK [ssl : Install ssl keys] *****
ok: [teamwiesn.com] => (item=teamwiesn.key)

TASK [ssl : Generate dhparam key] *****
ok: [teamwiesn.com]

TASK [ssl : Check ssl forward secrecy key permission] *****
ok: [teamwiesn.com]

TASK [ssl : include] *****
skipping: [teamwiesn.com]

TASK [pip : Check to see if pip is already installed.] *****
ok: [teamwiesn.com]
```

Es tracta d'un sistema basat en el servidor que s'executa en contenidors de servlets com Apache Tomcat. Admet eines de control de versions, incloent *AccuRev*, *CVS*, *Subversion*, *Git*, *Mercurial*, *Perforce*, *ClearCase* i *RTC*, i pot executar projectes basats en *Apache Ant*, *Apache Maven* i *sbt*, així com *scripts* de *shell* arbitraris i ordres de *batch* de *Windows*. El creador de *Jenkins* és Kohsuke Kawaguchi. Publicat sota la llicència MIT, *Jenkins* és programari lliure.

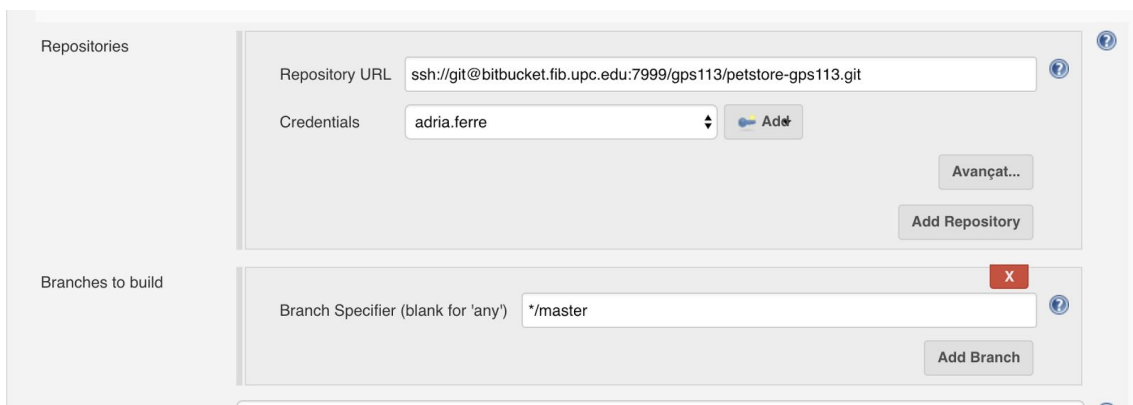
## Documentació

Amb el servidor *Jenkins* lo que voldrem fer serà una *pipeline*. Una *pipeline* es un grup d'events o jobs encadenats i integrats amb la resta de forma seqüencial. Aquesta constarà d'un conjunt de tasques, a les que anomenarem *jobs*. S'executarán una darrera l'altre en realitzar-se alguna acció establerta. Amb això aconseguirem una certa automatització i així mantenir en bon estat el projecte.

La *pipeline* constarà de 4 jobs: Build de l'aplicació, job de probes unitàries, job de SonarQube i Job Heroku.

## Build de l'aplicació

Primer de tot creem un nou job el qual farà tota la part del *Build* del nostre projecte. Hem posat la nostra URL del ssh del nostre projecte del bitbucket com a repositori a Jenkins, indicant la branca sobre la qual volíem treballar.

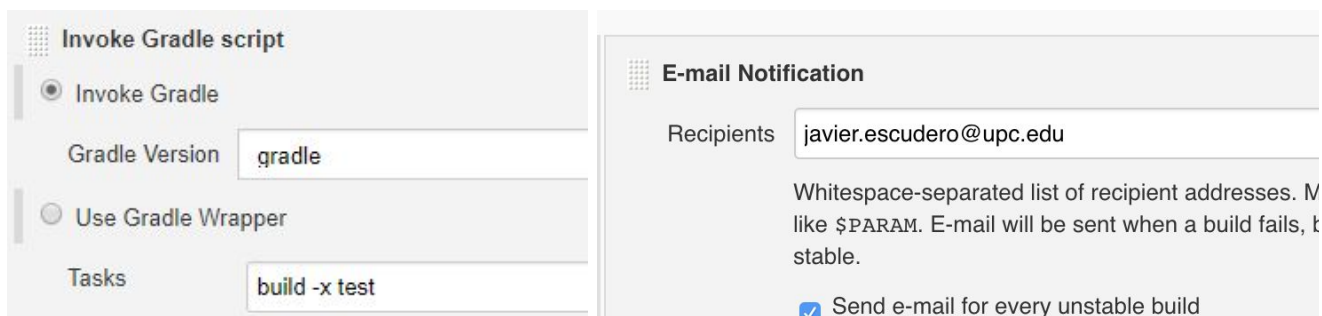


Amb els *Build triggers* s'indiquen unes condicions les quals si es compleixen es realitzarà un *build* del nostre projecte. Sabent això activarem, en els *Build triggers*, la opció de fer *build* cada cop que es realitza un push del nostre projecte al *Bitbucket*. També activarem el de poll SCM.

Build triggers>Build when a change is pushed to Bitbucket

Build triggers > Consultar repositorio (SCM)

Hem configurat les accions *post-build* per tal de que cada *build* el qual doni algun error ens envii un *mail* per tal de saber que algo està fallant.



## Job de probes unitàries

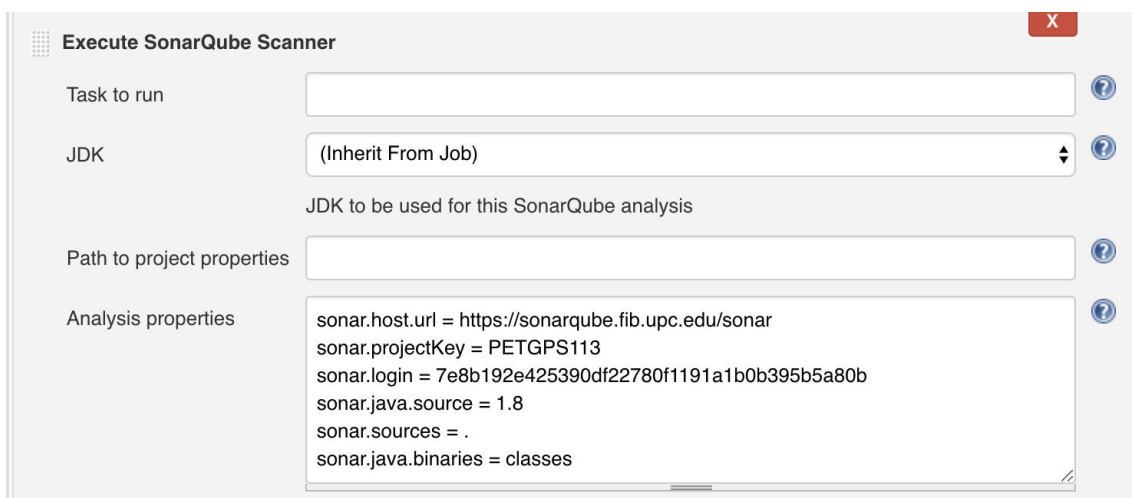
Un cop finalitzat el build de l'aplicació i comprovar el seu correcte funcionament procedim a la part de la realització dels test. Un cop modificat el *build.gradle* només haurem d'invocar-lo de manera que es realitzaran les nostres probes unitàries.

Modifiquem el fitxer del projecte *build.gradle* per problemes UTF-8.

```
7 allprojects {
8     apply plugin: 'java'
9     apply plugin: 'idea'
10
11     group = 'edu.upc.fib.petstore'
12     version = '16.10.26'
13
14     tasks.withType(JavaCompile) {
15         options.encoding = 'UTF-8'
16     }
17
18     tasks.withType(Test) {
19         systemProperty "file.encoding", "UTF-8"
20     }
21
22     repositories {
23         maven { url "https://oss.sonatype.org/content/repositories/releases" }
24         mavenCentral()
25     }
26 }
```

## Job de sonarqube

Creem un nou *job*, de l'apartat de *deploy*, el qual integrarà la nostra *pipeline* amb *SonarQube*. Activarem un plugin el qual permeti a Jenkins connectar-se fàcilment a *SonarQube*, així Sonar podrà executar el nostre codi, i nosaltres veure els code smells, *bugs*, *vulnerabilitats*... Per tal de configurar-ho haurem d'afegir la URL del servei, la versió i el *token* d'un usuari amb accés al projecte.



**Execute SonarQube Scanner**

Task to run:

JDK:

JDK to be used for this SonarQube analysis

Path to project properties:

Analysis properties: 

```
sonar.host.url = https://sonarqube.fib.upc.edu/sonar
sonar.projectKey = PETGPS113
sonar.login = 7e8b192e425390df22780f1191a1b0b395b5a80b
sonar.java.source = 1.8
sonar.sources = .
sonar.java.binaries = classes
```

Executarem l'*Scanner SonarQube* amb el *projectKey* i *sonar.login* corresponent al nostre grup.

Un cop instal·lat l'*scanner* haurem de fixar les prioritats d'aquest.

Un cop fet tot això comprovem que efectivament quan modifiquem algo al *Bitbucket* es fa *build* del projecte, passa els test i s'actualitza al *SonarQube*.

## PART OPCIONAL:



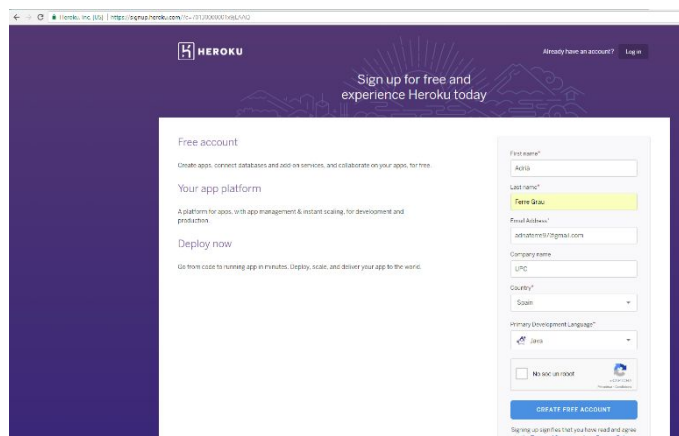
### Introducció

Heroku és una plataforma en el núvol com un servei (PaaS) que dona suport a diversos llenguatges de programació, s'utilitza com a model d'implementació d'aplicacions web. Heroku, una de les primeres plataformes en núvol, ha estat en desenvolupament des de juny de 2007, quan només va ser compatible amb el llenguatge de programació de Ruby, però ara és compatible amb *Java*, *Node.js*, *Scala*, *Clojure*, *Python*, *PHP* i *Go*. Per aquest motiu, es diu que *Heroku* és una plataforma políglota ja que permet al desenvolupador construir, executar i escalar aplicacions de manera similar en tots els idiomes. Heroku va ser adquirida per *Salesforce.com* el 2010 per 212 milions de dòlars.

### Job de Heroku

Lo que ens permetrà *Heroku* serà fer un release del nostre projecte.

Per tal de poder enllaçar Jenkins amb Heroku i així acabar la nostre *pipeline* primer ens hem de crear un compte a <https://signup.heroku.com/>

A screenshot of the Heroku sign-up page. The page has a dark purple header with the Heroku logo and the text 'Sign up for free and experience Heroku today'. Below the header, there's a white box containing the sign-up form. The form is titled 'Free account' and includes sections for 'Your app platform' and 'Deploy now'. The right side of the form contains input fields for 'First name', 'Last name', 'Email address', 'Company name', 'Country', 'Primary development language', and a checkbox for 'No social login'. At the bottom of the form is a blue button labeled 'CREATE FREE ACCOUNT'.

Un cop feta la nostre compte haurem d'instal·lar l'aplicació. Un cop instal·lada l'hem de configurar des de la consola:

```
> heroku create petstore-gps113 --buildpack heroku/gradle
```

```
> heroku config:set GRADLE_TASK="build --x test"
```

```
> heroku git:remote --ssh-git --app petstore-gps113.
```

Un cop enllestida la configuració hem omplert les taules mitjançant el script sql.

Actualment, al fer un push es realitzaran tots els jobs descrits anteriorment, i cada cop que es realitza un d'ells, al menú principal del Jenkins podrem veure en tot moment si aquestes execucions s'han aconseguit realitzar correctament, o si contràriament, ha hagut algun error, així com un historial per poder veure com ha sigut el transcurs de les execucions anteriors.