

Práctica 3: Integración, entrega y despliegue continuo

Ampliación de Ingeniería del Software

3º Curso - Grado Ingeniería Informática

Luis Miguel Jiménez Aliaga



Universidad
Rey Juan Carlos

GitFlow

El desarrollo de la nueva funcionalidad para la aplicación con GitFlow empieza a partir del commit “9bd8065”. Los commits anteriores son pruebas fallidas y desarrollo de los workflows necesarios para la práctica.

Primero he clonado el repositorio de GitHub a uno local y he creado la rama develop. Después he hecho push y se ha creado dicha rama en el remoto:

```
$ git clone https://github.com/lm-jim/ais-lm.jimenez-urjc-2021.git
Cloning into 'ais-lm.jimenez-urjc-2021'...
remote: Enumerating objects: 678, done.
remote: Counting objects: 100% (678/678), done.
remote: Compressing objects: 100% (230/230), done.
Receivemote: Total 678 (delta 199), reused 652 (delta 187), pack-reused 0
Receiving objects: 100% (678/678), 88.55 KiB | 1.15 MiB/s, done.
Resolving deltas: 100% (199/199), done.
```

```
$ git branch develop
```

```
$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/lm-jim/ais-lm.jimenez-urjc-2021/pull/new/develop
remote:
To https://github.com/lm-jim/ais-lm.jimenez-urjc-2021.git
 * [new branch]      develop -> develop
```

Después hago un proceso parecido, creando una rama desde develop llamada “feature/lineBreaker”. Esta es la rama en la que desarrollaremos la nueva funcionalidad.

```
$ git checkout develop
Switched to branch 'develop'
```

```
$ git branch feature/lineBreaker
```

```
$ git push origin feature/lineBreaker
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/lineBreaker' on GitHub by visiting:
remote:   https://github.com/lm-jim/ais-lm.jimenez-urjc-2021/pull/new/feature/lineBreaker
remote:
To https://github.com/lm-jim/ais-lm.jimenez-urjc-2021.git
 * [new branch]      feature/lineBreaker -> feature/lineBreaker
```

Ahora tengo 3 ramas de GitFlow en el repositorio local y el remoto. Me coloco en la rama de feature y añado el archivo LineBreaker.java que se proporcionan en el aula virtual.

```
$ git checkout feature/lineBreaker
Switched to branch 'feature/lineBreaker'
```

```
$ git status
On branch feature/lineBreaker
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    src/main/java/es/urjc/code/daw/library/book/LineBreaker.java

nothing added to commit but untracked files present (use "git add" to track)
```

Lo pongo en staged y hago un nuevo commit:

```
$ git add src/main/java/es/urjc/code/daw/library/book/LineBreaker.java

$ git commit -m "Added LineBreaker.java"
[feature/lineBreaker 7f1fdf1] Added LineBreaker.java
1 file changed, 63 insertions(+)
create mode 100644 src/main/java/es/urjc/code/daw/library/book/LineBreaker.java
```

Repito el mismo proceso para el test en un nuevo commit.

```
$ git commit -m "Added LineBreaker test"
[feature/lineBreaker 081f410] Added LineBreaker test
1 file changed, 123 insertions(+)
create mode 100644 src/test/java/es/urjc/code/daw/library/unitary/LineBreakerUnitaryTest.java
```

Pusheamos al repositorio remoto


```
$ git push origin feature/lineBreaker
Enumerating objects: 36, done.
Counting objects: 100% (36/36), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (21/21), 2.22 KiB | 759.00 KiB/s, done.
Total 21 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/lm-jim/ais-lm.jimenez-urjc-2021.git
   52bc5e8..c829176  feature/lineBreaker -> feature/lineBreaker
```



Al crear el pull Request en GitHub se ejecuta el workflow correspondiente:

(edito: no se ejecuta este workflow automáticamente. No he conseguido arreglarlo antes del momento de entregar la práctica, pero en teoría debería, ya que desencadena con el evento “pull_request”. De todas formas, el workflow funciona bien cuando se ejecuta manualmente.)

```
name: FeaturePullRequest

on:
  pull_request:
    branches: feature/*
  workflow_dispatch:
```


 **FeaturePullRequest**
FeaturePullRequest #5: Manually run by lm-jim


 1 minute ago
 53s

...


Ahora estamos listos para hacer un merge de la rama feature a la develop por medio de un pull request.



Merge lineBreaker with develop #23

 **Merged**

 lm-jim merged 2 commits into **develop** from **feature/lineBreaker**  now

Al hacer el merge se ejecuta el workflow de DevelopCommit (cuando hacemos un commit a la rama develop).

 **Merge pull request #23 from lm-jim/feature/lineBreaker**
DevelopCommit #12: Commit fd2eb25 pushed by lm-jim

develop
 2 minutes ago
 2m 1s

.

Los test se han ejecutado correctamente así que de momento no tenemos ningún problema. Ahora haremos un commit desde develop, aumentando el número de versión en el pom.xml, luego un pull request a la rama release, y quitaremos el SNAPSHOT en esa rama.

```
$ git branch release/0.1.0
```

```

GNU nano 5.4                                pom.xml                                Modified
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apac
  <modelVersion>4.0.0</modelVersion>

  <groupId>es.urjc.code</groupId>
  <artifactId>practica_1_testing</artifactId>
  <version>0.2.0-SNAPSHOT</version>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.1</version>
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>11</java.version>
  </properties>

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line

```

```

$ git commit -m "Increased version to 0.2.0"
[develop 6acd342] Increased version to 0.2.0
1 file changed, 1 insertion(+), 1 deletion(-)

```

```

$ git push origin release/0.1.0

```

Los workflows se ejecutan correctamente. Hacemos un nuevo commit en release para cambiar la versión quitando SNAPSHOT. Como las pruebas siguen ejecutando correctamente, podemos hacer un último merge con la rama master para sacar la release.

```

GNU nano 5.4                                pom.xml                                Modified
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org>
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apac>
  <modelVersion>4.0.0</modelVersion>

  <groupId>es.urjc.code</groupId>
  <artifactId>practica_1_testing</artifactId>
  <version>0.1.0</version>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.1</version>
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>11</java.version>
  </properties>

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line

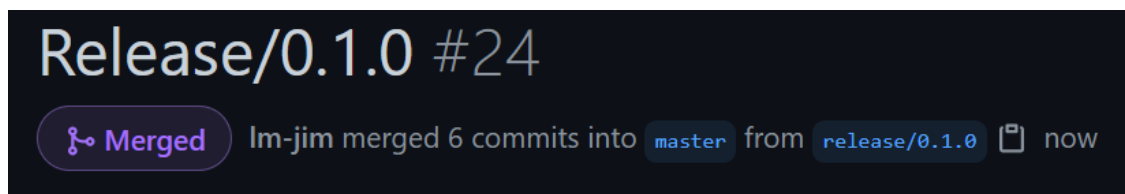
```

```
$ git commit -m "Removed SNAPSHOT from ver."
[release/0.1.0 979cd6d] Removed SNAPSHOT from ver.
1 file changed, 1 insertion(+), 1 deletion(-)
```

Todas las pruebas se han ejecutado correctamente. (Hay algún commit extra porque me equivoqué y cambié la versión antes de crear el branch release desde develop)

✓ Removed SNAPSHOT from ver. ReleaseCommit #6: Commit 979cd6d pushed by lm-jim	release/0.1.0	📅 1 minute ago 🕒 1m 2s	...
✓ Set version back to 0.1.0 ReleaseCommit #5: Commit 5a66b74 pushed by lm-jim	release/0.1.0	📅 4 minutes ago 🕒 55s	...
✓ Increased version to 0.2.0 DevelopCommit #15: Commit 6acd342 pushed by lm-jim	develop	📅 5 minutes ago 🕒 1m 50s	...
✓ Set version back to 0.1.0 DevelopCommit #14: Commit 5a66b74 pushed by lm-jim	develop	📅 8 minutes ago 🕒 2m 3s	...
✓ Set version to 0.2.0 DevelopCommit #13: Commit 4e24ca5 pushed by lm-jim	develop	📅 29 minutes ago 🕒 2m 4s	...

Lo último que nos queda es hacer merge desde develop a la rama master. Con ello habremos finalizado el proceso de GitFlow y tendremos una versión estable.



El último workflow se ejecuta correctamente y ya tenemos la versión nueva desplegada:

105 workflow runs				Event ▾	Status ▾	Branch ▾	Actor ▾
✓ Merge pull request #24 from lm-jim/release/0.1.0 MasterCommit #44: Commit 5970a77 pushed by lm-jim	master	📅 8 minutes ago 🕒 7m 52s	...				

Workflows

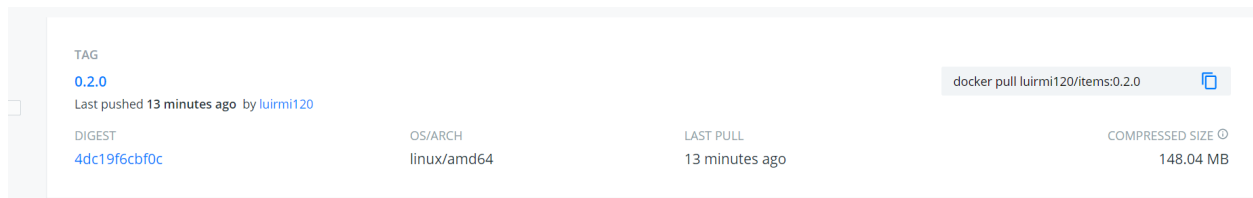
Para la práctica he desarrollado 5 archivos .yaml, para cada uno de los workflows que se pedían en la práctica:

Todos funcionan correctamente menos el de PullRequest, ya que no se ejecuta automáticamente. Sin embargo, al ejecutarlo manualmente da buenos resultados.

developCommit.yaml: Se ejecuta cuando se hace un push en la rama develop. Se ejecutan todos los test de la aplicación.

featurePullReq.yaml: Se debería ejecutar cuando se hace un pull request en las ramas feature. Se ejecutan los test unitarios y los de api REST.

masterCommit.yaml: Se ejecuta cuando se hace un commit en la rama master. Primero se ejecutan todos los test en local y se crea el artefacto, luego, en el siguiente job, se publica el mismo en DockerHub, con su nombre de version que le corresponde:





TAG			
0.2.0			docker pull luirmi120/items:0.2.0
Last pushed 13 minutes ago by luirmi120			
DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE
4dc19f6cbf0c	linux/amd64	13 minutes ago	148.04 MB

En los dos siguientes jobs se hace toda la configuración necesaria para desplegarlo en Heroku, y, en este último, se ejecutan los test de API REST y Selenium de nuevo, pero orientados a la aplicación desplegada. Es por eso que pasé como argumento parte de la URL a los test. El test de API Rest se pone por defecto en el puerto 443, ya que, en caso contrario no se realizaban las peticiones correctamente. En Selenium no es necesario.

nightly.yaml: Se ejecuta todos los días a las 2 AM (UTC), gracias al evento schedule:

```
on:
  schedule:
    - cron: '0 2 * * *' #2AM UTC
  workflow_dispatch:
```

Cuando se ejecuta, pasa todos los tests, crea un artefacto y lo sube a dockerhub, con un tag del tipo dev-”fecha”:

TAG				docker pull luirmi120/items:dev-07_06_2021 		
dev-07_06_2021						
Last pushed 4 hours ago by luirmi120						
DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE 			
ea2813905394	linux/amd64	4 hours ago	148.04 MB			

releaseCommit.yml: Se ejecuta cada vez que se hace un commit en ramas release. Ejecuta los test unitarios y de API Rest.

Para poder hacer compatibles los test, he tenido que modificar los archivos de pruebas API rest y Selenium, añadiendo la posibilidad de que el host no sea el local (para poder pasar los tests en la aplicación de Heroku)

URL del repositorio GitHub: [lm-jim/ais-lm.jimenez-urjc-2021 \(github.com\)](https://github.com/lm-jim/ais-lm.jimenez-urjc-2021)

URL de la aplicación Heroku: [Main \(ais-lmjimenez-2021.herokuapp.com\)](https://ais-lmjimenez-2021.herokuapp.com/)

URL de los deployments en Heroku: [ais-lmjimenez-2021 | Heroku](https://ais-lmjimenez-2021.herokuapp.com/deployments)