Control de versions

Projecte Integrat de Software

Universitat de Barcelona

16 de Març de 2022

Overview

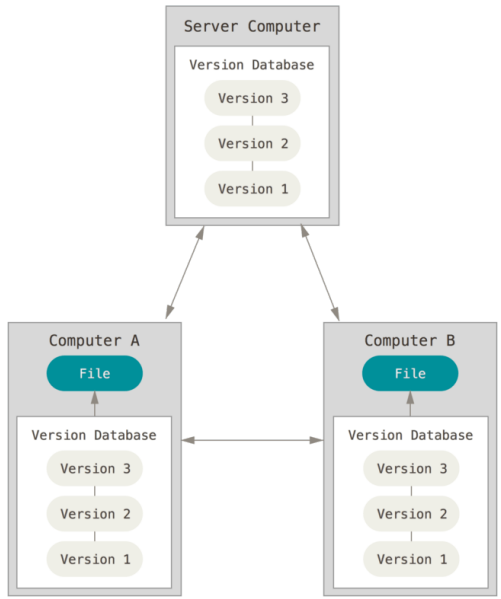
Git basics

- Git és un projecte open-source per fer control de versions de projectes de software, especialment indicat per treballs col·laboratius. És l'estàndard en desenvolupament de software (80% d'ús de mercat).
- És un *Sistema de Control de Versions Descentralitzat*, el que significa que el projecte es copia de forma íntegra amb tots els canvis fets des de la seva creació.
- Alguns exemples de serveis per mantenir una còpia d'aquests repositories són GitHub, GitLab o Bitbucket.

GitHub GitLab Bitbucket

git

git-scm.com/book



Crear un repositori

Podem fer-ho mitjançant la llibreria de la terminal *git* o un entorn gràfic, com GitKraken (o Android Studio). Els següents exemples fan servir la línia de comandes.

- Per crear un repositori des de zero fem: `git init`
  Crea la carpeta **.git**, que conté la metadata del repositori.
- Per afegir un nou element al repositori, creem el fitxer (p.e. *file.txt*) i executem les comandes:
  `git add file.txt`
  `git commit -m "Description of changes"`

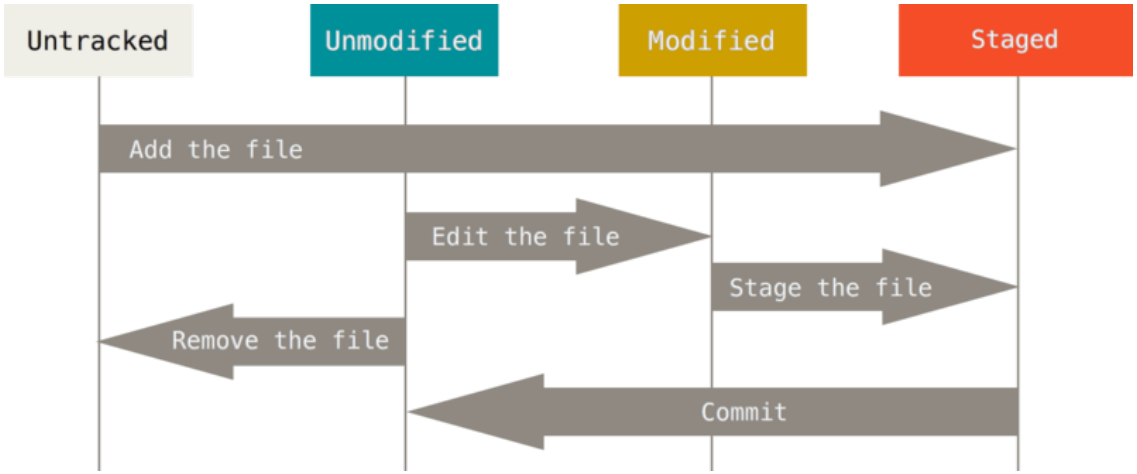
Si el repositori ja existeix i està disponible en un altre ordinador, podem executar

`git clone <url>`

per fer una còpia local.

Guardar canvis al repositori

- Com hem vist abans, podem afegir elements nous amb les comandes *add* i *commit*. Podem veure l'estat dels nostres fitxers del repositori amb: `git status`
- Aquests són diferents estats i com es passa d'un a l'altre



- Els canvis quedaran guardats al repositori al fer `git commit`
- Hi ha l'opció d'ignorar fitxers per a que no apareguin al cicle anterior. Això es fa amb el fitxer **.gitignore**.

Exemple de .gitignore

```
# ignore all .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
!lib.a

# only ignore the TODO file in the current directory, not subdir/TODO
/TODO

# ignore all files in any directory named build
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory and any of its subdirectories
doc/**/*.pdf
```

Podeu trobar .gitignore per a tot tipus de projectes a github.com/github/gitignore (també per a Android).

Veure canvis al repositori

- Per veure els canvis que no estan encara a *staged* es fa servir la comanda `git diff`

```
diff --git a/notebooks/Results.ipynb b/notebooks/Results.ipynb
index 0ae2b51..e475de0 100644
--- a/notebooks/Results.ipynb
+++ b/notebooks/Results.ipynb
@@ -67,17 +67,7 @@
    "execution_count": 4,
    "id": "valued-sympathy",
    "metadata": {},
-   "outputs": [
-    {
-     "name": "stderr",
-     "output_type": "stream",
-     "text": [
-      "/home/vec/miniconda3/envs/genv2/lib/python3.7/site-packages/outdated/uti
-      "Set the environment variable OUTDATED_IGNORE=1 to disable these warnings
-      "  **kwargs\n"
-     ]
-    }
-   ],
+   "outputs": [],
    "source": [
     "def step_grouping(datafr, gap=5):\n",
     "    datafr['big_step'] = 0\n",
@@ -146,7 +136,17 @@
    "execution_count": 5,
    "id": "decreased-junior",
```

- Ho fareu servir continuament, tot i que directament a Android Studio i no a la línia de comandes.

Repositoris remots

- Els repositoris remots són versions del projecte que estan mantinguts al web, per a facilitar l'accés a ells.
- Per col·laborar de forma eficient amb altres membres d'un projecte cal tenir aquests repositoris remots actualitzats.
- Com s'actualitzen? *Fetch*, *merge*, *push*, *pull*
- Per descarregar els canvis del repositori remot, farem `git fetch <remote>`. Veurem els canvis que s'han produït però no canviarà els nostres fitxers.
- Per incorporar els canvis als nostres fitxers: `git merge`.
- Els dos passos anteriors es poden fer en una sola comanda amb `git pull`.
- Per actualitzar els canvis locals al repositori remot fem `git push <remote> <branch>`.

Resum

Ja sabem com fer canvis a un repositori:

- Crear o clonar un repositori.
- Guardar canvis.
- Treballar i sincronitzar el repositori amb un de remot.

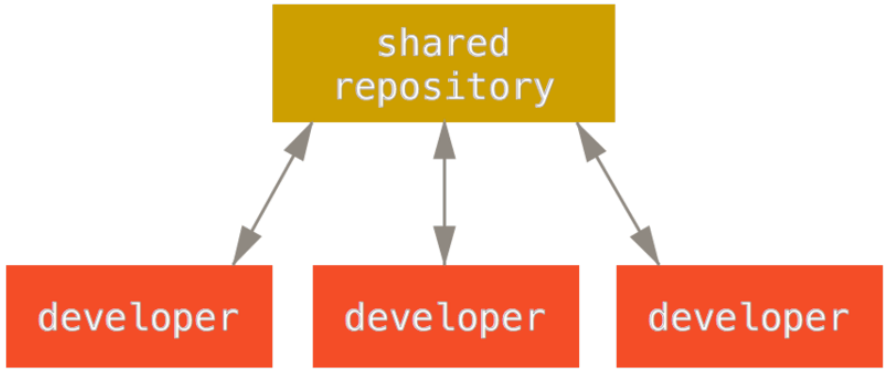
Branques

- La creació de branques o divergències és una de les característiques més importants de Git.
- Permet treballar amb diverses versions del projecte i mouren's d'una a l'altra ràpidament i de forma controlada.
- Per fer-ho, fem servir `git checkout <branch>` .

1. Git basics
2. Workflow
   - Example de flux de treball: John i Jessica
3. Git a Android

Flux de treball

El flux habitual, i amb el qual treballareu al vostre projecte, és un flux centralitzat:



Hi haurà un repositori "centralitzat" a un servidor i tots hi traballarem simultàniament amb una còpia.

John i Jessica

Vegeu el següent exemple, que il·lustra com contribuir a un repositori. Hi ha dos desenvolupadors, John i Jessica. Comencen per clonar-se el repositori existent i modificar o crear fitxers.

1. John modifica un fitxer.

```
# John's Machine
$ git clone john@githost:simplegit.git
Cloning into 'simplegit'...
...
$ cd simplegit/
$ vim lib/simplegit.rb
$ git commit -am 'remove invalid default value'
[master 738ee87] remove invalid default value
 1 files changed, 1 insertions(+), 1 deletions(-)
```

John i Jessica

2. Jessica modifica un altre fitxer i el puja al repositori remot.

```
# Jessica's Machine
$ git clone jessica@githost:simplegit.git
Cloning into 'simplegit'...
...
$ cd simplegit/
$ vim TODO
$ git commit -am 'add reset task'
[master fbff5bc] add reset task
 1 files changed, 1 insertions(+), 0 deletions(-)
```

```
# Jessica's Machine
$ git push origin master
...
To jessica@githost:simplegit.git
   1edee6b..fbff5bc  master -> master
```

John i Jessica

3. Si John intenta pujar els canvis ara, es trobarà un error. La comanda *push* només permet pujar els canvis si tens tots els canvis actualitzats amb el remot (els canvis de Jessica).

```
# John's Machine
$ git push origin master
To john@githost:simplegit.git
 ! [rejected]        master -> master (non-fast forward)
error: failed to push some refs to 'john@githost:simplegit.git'
```

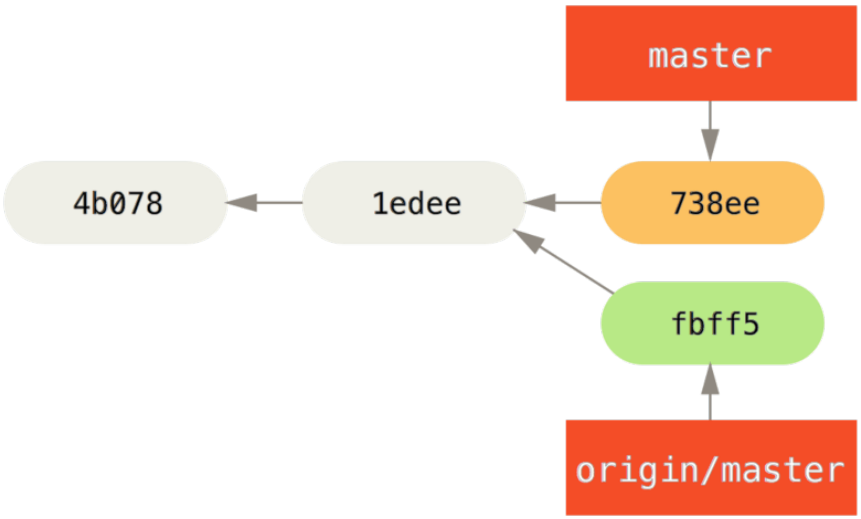
Git no permet combinar els canvis en remot. S'han de combinar localment abans de pujar-los.

John i Jessica

4. John ha de descarregar-se els canvis de Jessica, combinar-los amb els seus i pujar-ho tot al remot.

```
$ git fetch origin
...
From john@githost:simplegit
 + 049d078...fbff5bc master     -> origin/master
```

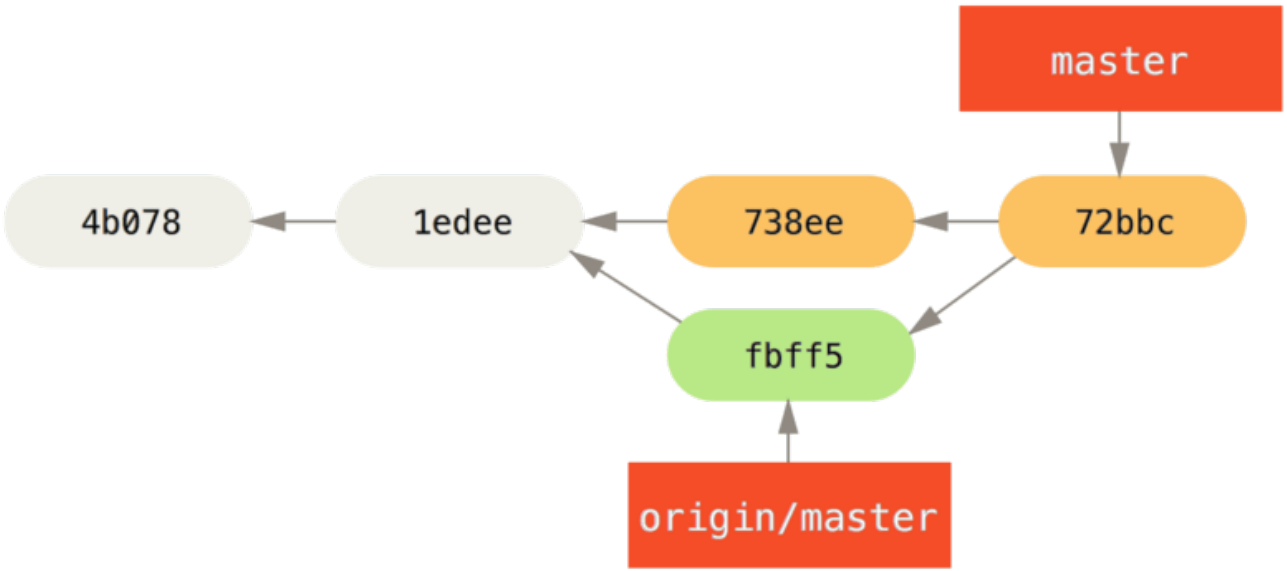
En aquest moment, per a John, el repositori te la següent configuració:



John i Jessica

5. Finalment, John combina els canvis dels dos commits (el seu i el de Jessica). Aquest pas i l'anterior es solen fer de forma automatitzada amb la comanda *pull*.

```
$ git merge origin/master
Merge made by the 'recursive' strategy.
 TODO |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
```



John i Jessica

5. Una vegada combinat el local i el remot, John pot sincronitzar el projecte amb el repositori remot.

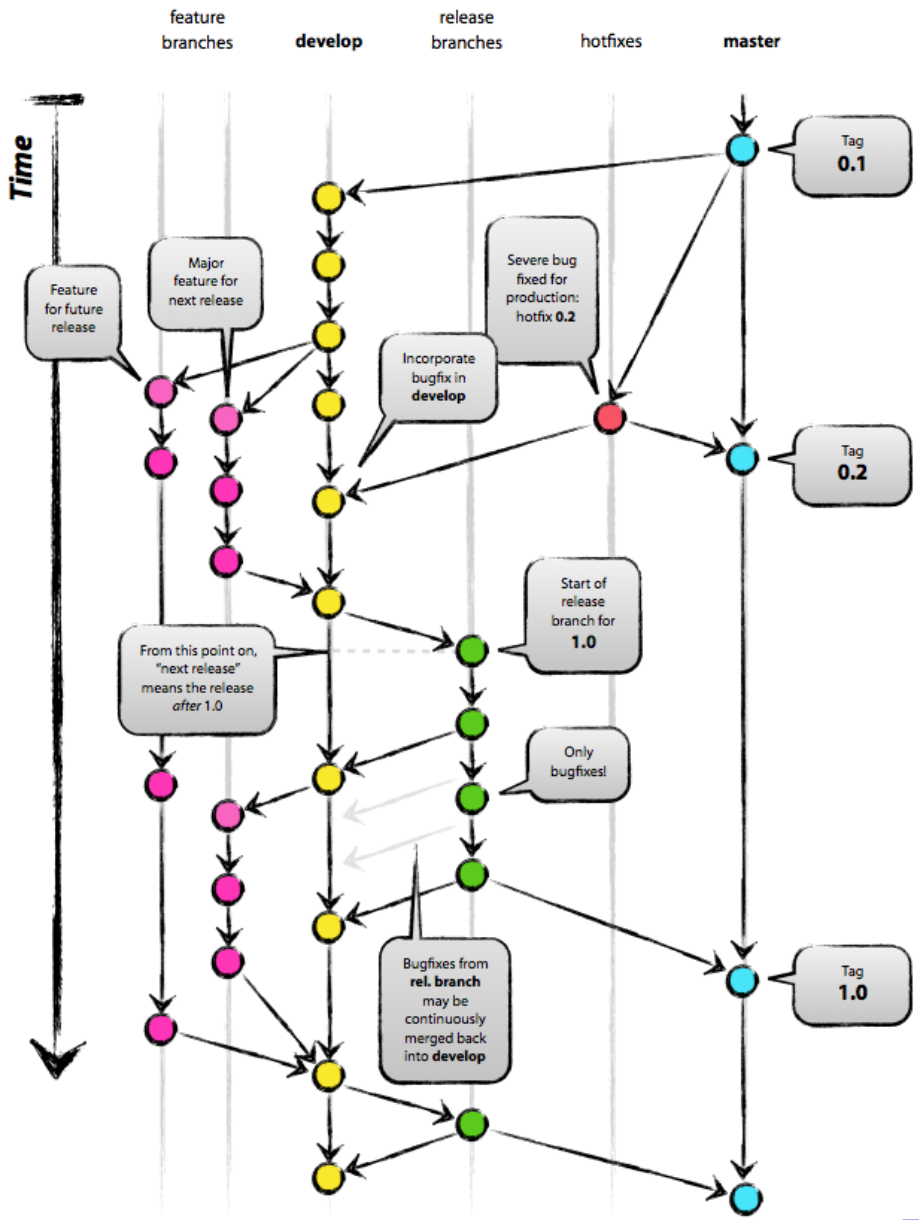
```
$ git push origin master
...
To john@githost:simplegit.git
   fbff5bc..72bbc59  master -> master
```

A la pràctica

- A la pràctica, el procés habitual hauria de ser el de John i Jessica, afegint alguna branca més si fos necessari.
- Un exemple de flux de treball real (amb tota mena de detall) es pot trobar al següent link:

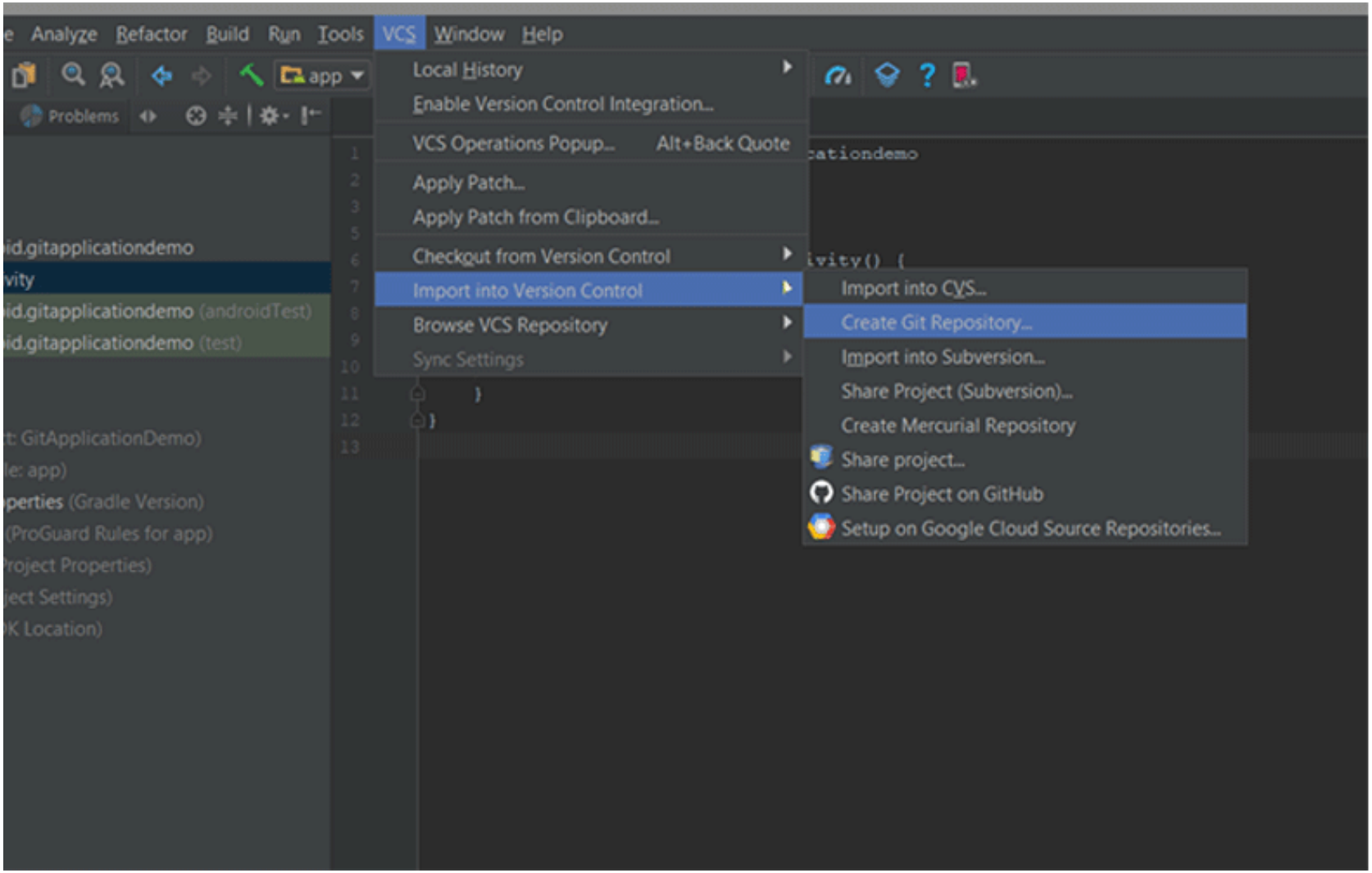
  nvie.com/posts/a-successful-git-branching-model/

A la pràctica



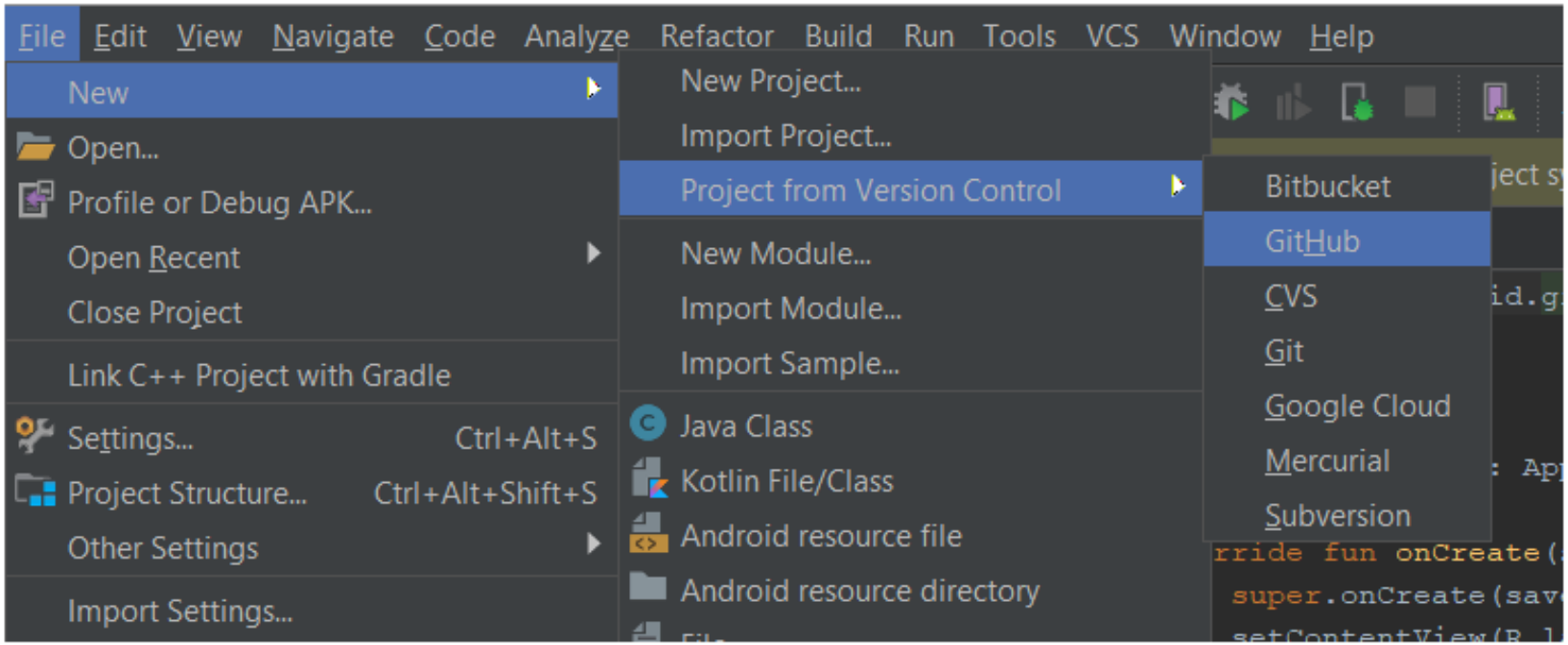
Crear repositori

- Crear un repositori d'un projecte existent.



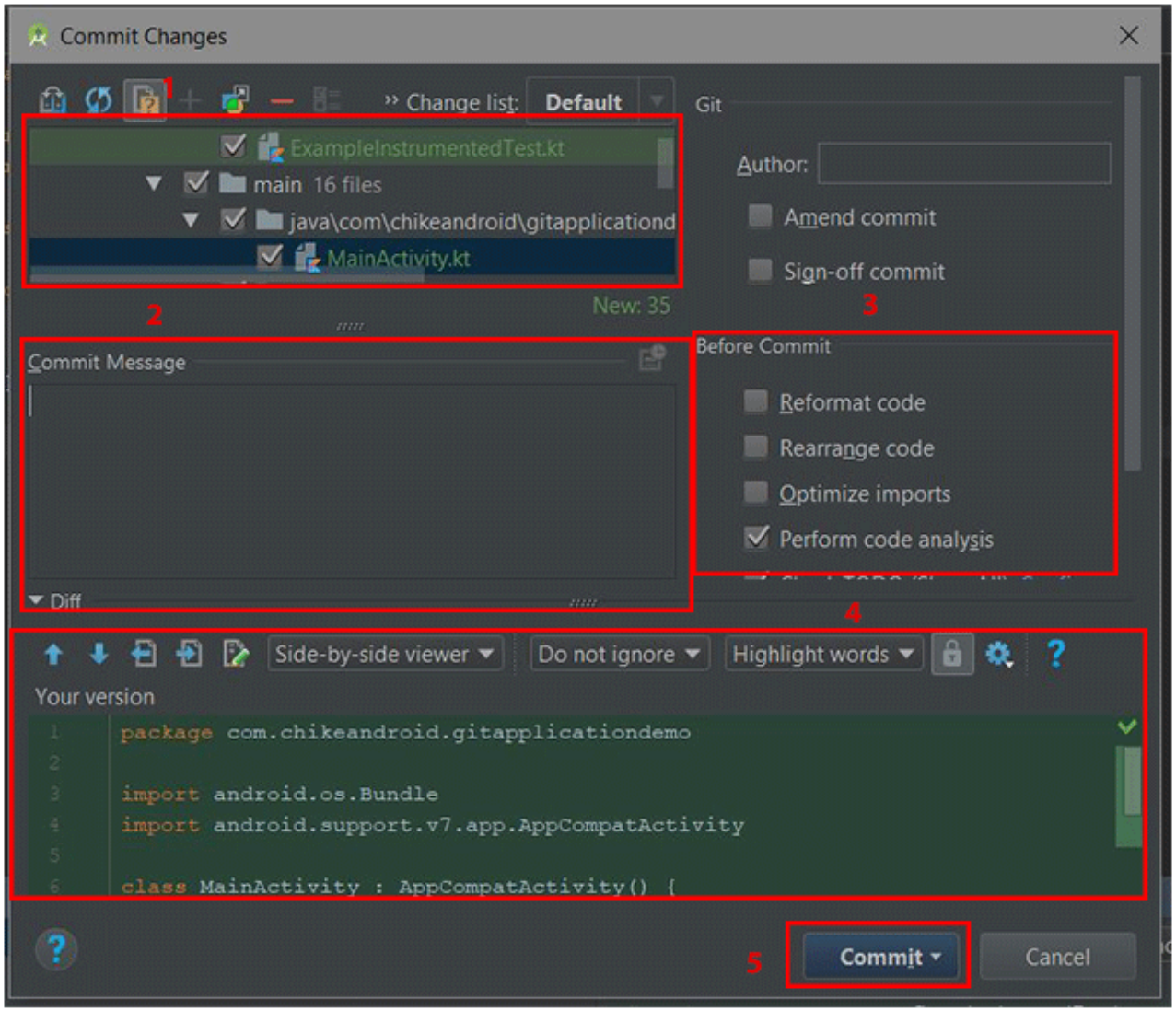
Importar repositori

- Importar un repositori del web.



Commits

- Guardar canvis.



Exemples de use cases per a l'entregable

Identificador	UC3 - Crear reserva
Descripción	Usuario crea una reserva y se guarda en la base de datos
Actores	Usuario registrado
Pre-condición	Usuario debe estar registrado y con la sesión iniciada
Flujo principal	1. Usuario entra en la primera página del proceso de reserva 2. Usuario selecciona fecha y deporte 3. Usuario selecciona pista y hora 4. Usuario especifica el numero de personas 5. Se añaden los datos a la base de datos 6. La aplicación redirige al usuario a la página principal
Flujo alternativo	2a. La aplicación comunica error si no se ha seleccionado fecha 2b. Usuario modifica los datos erróneos 4a. La aplicación comunica error si no se ha especificado personas 4b. Usuario modifica los datos erróneos
Post-condición	Reserva añadida a la base de datos y usuario vuelve a la página principal

Identificador	UC7 - Visualizar perfil
Descripción	Visualiza el perfil del usuario
Actores	Usuario registrado
Pre-condición	Usuario debe estar registrado y con la sesión iniciada
Flujo principal	1. 1. Usuario entra en la página de "Mi perfil" 2. La aplicación muestra una lista con datos del usuario
Flujo alternativo	No hay
Post-condición	Mostrada en la página "Mi perfil" los datos del usuario