Mini manual GitLab

1. Minicurs interessant:

Curs senzill de git[1]

- Començar

Per treballar localment amb un projecte Git primer de tot **és necessari clonar**(copiar)-lo localment.

Clonant un projecte del repositori.

```
git clone https://gitlab.com/nomUsuari/nomProjecte_X.git
git clone https://gitlab.com/nomUsuari/nomProjecte_Y.git
```

- Per pujar un projecte inicialment:

```
git add .
git commit -m "Commit inicial a data: `date +'%Y-%m-%d %H:%M:%S'`"
git push -u origin master
```

- Per baixar canvis d'un projecte:

On:

- **REMOTE**: és el repositori remot, normalment *origin*.
- **name-of-branch**: és la branca a baixar, normalment *master*.

```
git pull <REMOTE> <name-of-branch>
git pull origin master
```

2. Per veure tots els repositoris remots:

```
git remote -v
```

[1] https://learngitbranching.js.org/?locale=es_ES

3. Per afegir un enllaç a un repositori remot.

```
git remote add <source-name> <repository-path>
```

4. Per crear una branca sense interferir amb la màster.

```
git checkout -b <name-of-branch>
```

5. Per anar a una branca de treball

```
git checkout <name-of-branch>
```

6. Per veure els canvis que has fet

```
git status
```

7. Per veure les diferències entre la versió local i el repositori

```
git diff
```

8. Per guardar els canvis fets al projecte

```
git add <nom_arxiu-O-nom_carpeta>
git commit -m "comentari del commit"
```

9. Per guardar els canvis fets a tot el projecte

```
git add .
git commit -m "comentari del commit"
```

10. Per guardar els canvis fets, baixant primer allò que tenim pendent de baixar.

Primer en hem de situar al directori del projecte.

```
git pull origin master
```

```
git add .
git commit -m "add SOMETHING"
git push -u origin master
```

11. Enviar els canvis al repositori remot

```
git push <remote> <name-of-branch>
```

12. Crear un projecte buit al repositori i pujar-hi tot un projecte ja creat.

Configuració global.

```
git config --global user.name "Nom Usuari"
git config --global user.email "correu@domini.cat"
```

Creant un nou repositori TEST.

```
git clone https://gitlab.com/usuari/test.git
cd test
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Pujant una carpeta existent.

```
cd existing_folder
git init
git remote add origin https://gitlab.com/usuari/test.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

13. Comandes bàsiques

IMPORTANT:

- **commit**: serveix per fer fotos del moment exacte del nostre repositori local al remot

- **branch [nom]**: serveix per tal de crear una branca del projecte
- **checkout [nom]**: serveix per canviar de branca
- **merge [origen]**: serveix per fer un commit de dos pares i els seus antecessors en una branca. Per tal que les dues branques siguin igual hauríem de fer un segon *merge* per tal d'igualar les dues branques
- **rebase [origen]**: per combinar seqüencialment la feina de branques diferents. Per fer avançar, també la branca màster haurem de fer un altre *rebase*

14. Comandes per moures per les branques

- **HEAD**: apunta al commit més recent. Normalment **HEAD** apunta al nom d'una branca. Quan crees un *commit*, l'estat de la branca s'altera i aquest canvi és visible mitjançant el **HEAD**
- **Desadjuntar un HEAD**: acció d'adjuntar el **HEAD** a un *commit* en comptes de la branca. Per fer-ho utilitzare le *checkout* sobre el *hash* del *commit*
- *git log*: per tal de veure els *hash* dels *commit*

Referències relatives:

- ^: per moure's un *commit* enrera
- ~<*num*>: per moure's *n commit's* enrera

Exemples de canvis de HEAD amb referències relatives:

```
git checkout master^
git checkout master^^
git checkout master~2
```

15. Forçar branques

Per a reassignar una branca a un commit utilitzarem la opció *-f*.

Exemple de canvi forçat de branca:

```
git branch -f master HEAD~3
```

16. Revertint canvis al *git*

- *git reset*: desfà els canvis movent la referència d'una branca cap a endarrera en el temps a un *commit* anterior. Es pot imaginar com "reescriure la història". **Només** és per branques locals
- *git revert*: per tal de revertir i compartir aquest canvi amb el repositori remot

17. Moure el treball

- *git cherry-pick* <*commit1*> <*commit2*> <*…*>: còpia una sèrie de *commits* sobre la posició actual (HEAD). Podem deplaçar uns *commit's* concrets (C2 i C4 per exemple) a la posició actual

18. *rebase* interactiu

- *git rebase -i*: incloent aquesta opció, git mostrarà quins *commit* s'estan a punt de copiar sobre l'objectiu del *rebase*, els seus *hash* i els missatges

Quan s'obre el *rebase* interactiu es poden fer 3 coses:

- Reordenar els *commit* només canviant-ne l'ordre
- Ignorar completament algun *commit*. Si no s'assigna l'ordre *pick* a un *commit* vol dir que es vol ignorar
- Combinar diversos *commit* en un

19. Fent malabars amb els *commit*

19.1. Introduir petits canvis i reordenar els commit

- *git commit --amend*: s'utilitza per aplicar petits canvis d'un *commit* i s'utilitza conjuntament amb les comandes *rebase -i* i *cherry-pick*

19.2. Posar tags als commit

- *git tag etiqueta* <*commit*>: a vegades cal etiquetar un *commit* important, que no cal que sigui l'inici d'una nova branca. Per exemple: la finalització d'una tasca en concret. Si no li indiquem a quin *commit* volem l'etiqueta la posarà per defecte al *commit* que tingui el *HEAD*.

Exemple d'ús:

```
git tag fiTasca_x Cx
```

- *git describe <ref>*: on *ref* és la branca de la que volem informació. Ens diu a quina distància estem del *tag* més pròxim i quin *hash* de l'arbre ens trobem

20. Git Remot

Repositori remot: si tenir un repositori local ja és bo, tenir-lo remot ens proporciona la seguretat que si tenim un problema greu en el nostre repositori local, sempre podrem accedir al repositori remot. Una de les altres grans avantatges és que amb un repositori remot es pot compartir el codi.

20.1. Crear un repositori remot

- *git clone*: crea un repositori local a partir d'un repositori remot

20.2. Branques remotes

Les branques remotes tenen la particularitat que tenen un HEAD independent. Fins que no sincronitzeu les branques locals amb la remota, aquest no s'actualitzarà.

Les branques remotes s'han d'anomenar de la següent manera:

```
<remote name>/<branch name>
```

Normalment el nom remot del repositori serà anomenat *o* o *origin*.

20.3. Recuperar dades del repositori remot

- *git fetch*: recupera dades del repositori remot. És a dir que sincronitza la representació local del repositori amb el veritable estat del repositori remot

  # Baixa els *commits* que hi ha al repositori, però encara no en local

  # Actualitza on apunten les branques remotes

  # **No** actualitza l'estat actual de la branca local

- *git pull*: recupera les dades del repositori remot i a més a més actualitza l'estat actual de la branca local. Bàsicament és el mateix que fer un *fetch* i un *merge* consecutivament

20.4. Publicar dades al repositori remot

- *git push*: puja els canvis que s'han fet en local al repositori remot sincronitzant els estats. No permet per la publicació de canvis si algú ja n'ha fet i entren en conflicte amb el nostre
- *git push --rebase*: actualitza el repositori local fent els canvis que calen per poder-lo sincronitzar amb el remot

20.5. Bloqueig de la branca master

Treballant amb equips col·laboratius és recomanable tenir la branca *master* bloquejada i que es necessiti demanar permís per per un *push*.