**Comandos de Git y Bash**

**Basicos de bash:**

history

ls/dir: [-al]

cat: [file]

touch: ‘name of file’

pwd

cd

alias –‘name’=’variable’

**Basicos de git:**

git init

git add: [file] , [.]

git status

git config: [--list] , [--help]

git commit: [-m ‘ ”texto” ’] , [-am ‘ “texto” ’]

git log: [file] , [--all] , [--graph] , [--decorate] , [oneline] , [--stat]

git diff: [(codigo de commit codigo de commit)]

git show: [file]

git reset: [codigo de commit] [--hard]

[--soft] : guarda lo que esta en la zona de staying

git checkout: [codigo de commit] [rama]

git rm –cached: [file]

git restore --staged: [file]

git branch: [rama] . [-D rama]

git show\_brach: [--all]

git merge: [(rama rama)]

git tag: [-a ‘nombre de version’ -m 'texto']

**Conexión con Git Hub:**

git pull: [(‘origin’ master)]

git push: [(‘origin’ master)]

git remote: [-v] , [(set-url ‘origin’ ‘url con ssh’]

**Llaves Publicas y Privadas:**

eval $(ssh-agent -s)

~ (en codigo ascii: alt+126)

ssh-add ~/.ssh/id\_rsa

**Git stash:** La idea es que tengo cambias en una rama que no están todavía commiteados y entonces no puedo hacer checkout a otra rama. Entonces git stash guarda temporalmente los cambios existentes y ya no hay cambios que añadir, por lo tanto, podes visitar otras ramas.

git stash : [list]

[pop] :‘aca se recupera lo ‘stasheado’’

[(branch ‘nombre de rama’)] : ‘crea una rama con lo stasheado’

La forma correcta de hacerlo es hacer el stash en master, después hacer el stash branch desde master, después \*\*\*commitear desde la nueva rama\*\*\*(importante, sino se pierde lo hecho en master), y después ya hacer checkout a master.

[drop] : ‘elimina el stash guardado’

**Git clean:** Es para borrar archivos que no han sido añadidos. Ignora los archivos que están en .gitignore.

git clean: [-dry-run] : muestra que se borraría

[-f] : borra los archivos

(esto no tiene nada que ver, pero si se desease borrar un directorio, es con rm –r ‘nombre del directorio’)

**Cherry pick:** permite, en vez de hacer merge de una rama a tu rama actual (en tal caso te traerías toda la rama), traerte un commit pasado de esa rama.

git cherry pick : [commit hash]

**Commit amend:** Para pegar los cambios al commit anterior sin hacer otro nuevo usamos git commit --amend Debe haberse hecho un git add antes.

**Git reflog:** permite saber mas que un simple git log. Permite ver los head que se fueron muriendo a través del tiempo y todas las cosas que no se ven en log.

**Git grep y log:** permite buscar cadenas de texto en los archivos

git grep :

[‘texto’]

[(-n ‘texto’)] : te dice la linea en que aparece

[(-c ‘texto’)] : te dice cuantas veces aparece y en que archivos

git log –S ‘texto’ : busca en los textos de los commits

**Otros:**

git shortlog -sn = muestra cuantos commit han hecho cada miembros del equipo.

git shortlog -sn --all = muestra cuantos commit han hecho cada miembros del equipo hasta los que han sido eliminado

git shortlog -sn --all --no-merge = muestra cuantos commit han hecho cada miembros quitando los eliminados sin los merges

git blame ARCHIVO = muestra quien hizo cada cosa linea por linea

git COMANDO --help = muestra como funciona el comando.

git blame ARCHIVO -Llinea\_inicial,linea\_final= muestra quien hizo cada cosa linea por linea indicándole desde que linea ver ejemplo -L35,50

\*\*git branch -r \*\*= se muestran todas las ramas remotas

git branch -a = se muestran todas las ramas tanto locales como remotas

para agregar alias globales: git config –global alias.aliasname ‘comando’