Tutorial GIT

Un sistema de control de versións

Introdución

- Unha versión dun software é o estado no que se atopa un determinado proxecto software nun momento concreto.
- O control de versións consiste en controlar os distintos cambios que ocorren no cambio de versións
- Debe proporcionar:
  - Mecanismo para almacenar os elementos xestionados.

- Permitir realizar cambios nos elementos almacenados.
- Rexistro histórico dos cambios realizados nos elementos.

- Hai moitos software que se encarga disto, entre eles:
  - CVS
  - Subversion
  - Git
  - Mercurial

Git

- En debian atopase no paquete git-core
- Función mínima:
  - Ler un repositorio, para facer unha copia local.
  - Ex:
    - git clone git://git.kernel.org/pub/scm/git/git.git
    - git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git

Crear repositorio local

- Pártese da situación inicial, na que temos un proxecto software dentro do noso equipo.
- Todos os arquivos do proxecto atópanse nun directorio do disco. Ex: proxecto1
- Primeiro é necesario inicializa-lo repositorio
  - Introducímonos dentro do directorio do proxecto
  - git init
  - Esto crea un directorio oculto chamado .git, que vai ser onde se almacena toda a información do control de versións

- Neste momento o que temos é un repositorio baleiro.
- Hai que introducir todos os arquivos do proxecto no repositorio
  - git add .
  - Isto introducirá tódolos arquivos do directorio actual no repositorio.
- Unha vez feito isto, fixar o estado do proxecto facendo un commit no repositorio

- git commit
- Git vainos preguntar datos sobre o commit. Esto quedará gardado xunto o demais.

Modificacións

- No traballo normal dentro do proxecto, vanse facendo modificacións
  - Modificase código de arquivos existentes.
  - Créanse arquivos novos
  - Elimínanse arquivos
- Para ver as diferencias entre o repositorio e os arquivos que temos actualmente no disco:
  - git status

- Unha vez que se sabe os arquivos novos e modificados temos que salvar eses cambios no repositorio
  - git add file1 file2 file3
  - Se o arquivo xa existía garda os cambios entre o arquivo orixinal e o actual
  - Se o arquivo é novo copiase no repositorio
- Terminado de indicar os cambios:
  - git commit
  - Fai os cambios permanentes

- Se non existen arquivos novos, e tan so hai modificacións de arquivos pódese usar:
  - git commit -a
  - Encargase de buscar os arquivos cambiados, engadir eses cambios o repositorio e face-lo commit. Todo en un.
-

Log

- O log permítenos comprobar os cambios que se fan no proxecto
  - git log
  - Mostra tódolos commit que se fixeron no proxecto, coa súa descripción
  - Git log –stat –sumary
  - Mostra un resumo dos commit realizados, con estadísticas: liñas cambiadas, eliminadas etc.

Manexo de ramas

- Unha rama é unha variación que se fai no código por algún motivo:
  - Probar unha idea nova
  - Facer algunha modificación para un cliente.
  - Facer unha nova versión
  - Por que si.
- Cada unha das ramas que se crea pode evolucionar independentemente.
- De principio existe unha unica rama: master

- Para lista-las ramas existentes:
  - git branch
  - Cun asterisco indica a rama seleccionada actualmente
- Para crear unha nova rama:
  - git branch prueba
  - O contido orixinal de prueba é igual o de a rama que estaba seleccionada anteriormente

- Para cambiarnos a nova rama:
  - git checkout prueba
- A partires deste momento pódese facer os cambios que se queira os arquivos
  - Cando se termina de facer as moficiacións faise un commit dos datos.
- Podese volver a rama master (ou calquera outra)
  - git checkout master

  - Os arquivos volven a te-lo contido que tiñan antes de face-las modificacións.
  - Os cambios quedaron almacenados no repositorio
- A rama master ou calquera outra pódese seguir modificando, sen afectar o que se fai noutras ramas.
- Pódese fusionar dúas ramas:
  - git merge prueba
  - Se non existe un conflito (as mesmas liñas modificadas nas dúas ramas) os cambios se mezclan sen problemas

- Se hai problemas deixará marcas no código que se poden ver con git diff
- Os conflitos hai que solucionalos a man.
- Cando se termina de resolver conflitos git commit -a (coma sempre)

- Pódese eliminar unha rama:
  - git branch -d prueba
  - Primeiro comprobará que os cambios feitos en prueba están incluídos na rama actual, senón dará un erro

- Se os cambios feitos nunha rama son desechados:
  - git branch -D prueba

Repositorios remotos

- Para poder realizar un traballo en grupo é imprescindible que todos teñan acceso ao código
  - Servidor remoto
- Teremos un servidor que teña acceso SSH
  - Nos vai a permitir logearnos e facer accesos o sistema de ficheiros
- Primeiro: determinar onde gardar os datos dos proxectos no servidor. Ex: /opt/git

- Segundo, hai que ter acceso ao servidor a través de SSH
- Ollo cos permisos das carpetas onde se gardan os datos

Creando proxecto baleiro (remoto)

- Nos poñemos no directorio onde se gardan
  - cd /opt/git
- Se crea unha carpeta para gardar o proxecto
  - mkdir meuproxecto.git
  - cd meuproxecto.git
- Se crea o proxecto baleiro
  - git --bare init
  - git config core.sharedRepository true

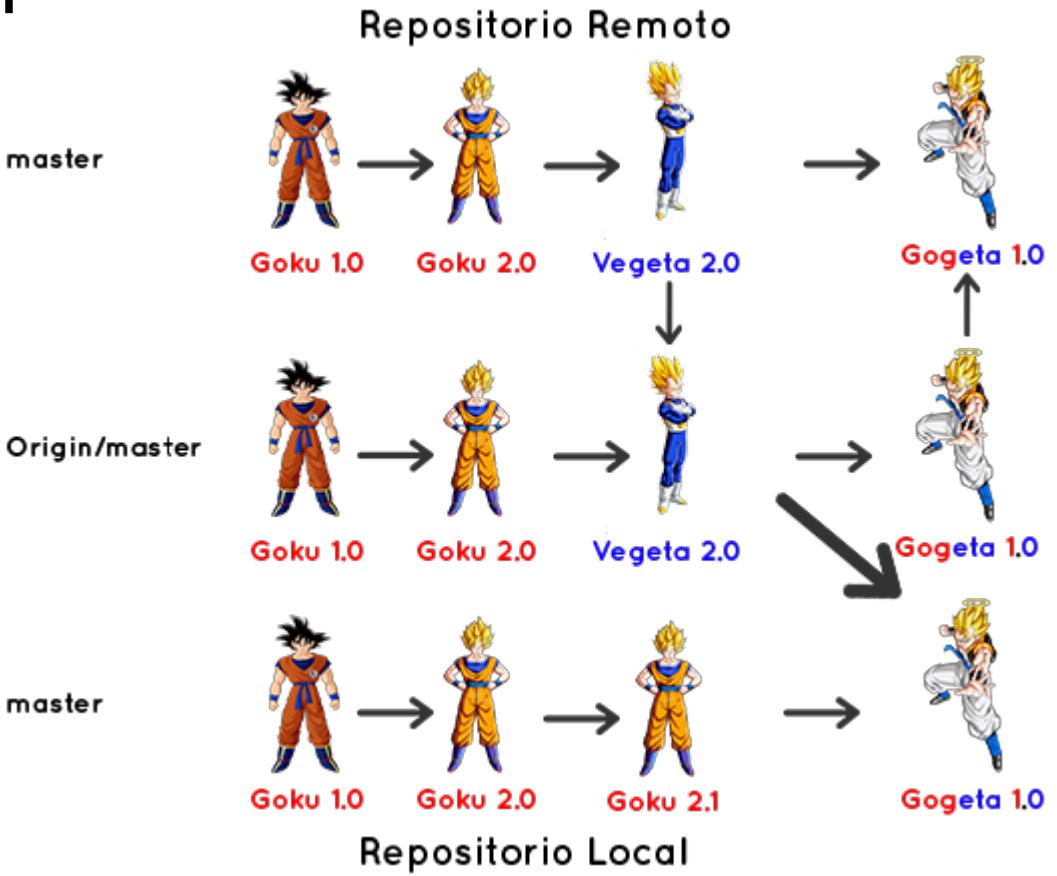
Creando repositorio (local).

- Fanse os pasos para crear un proxecto (git init , git add, git commit)
- Engadimos un orixe remoto de datos
  - git remote add **origin** git+ssh://user@server/opt/git/meuproxecto.git
  - origin, para indicar cal e o servidor base
- Para subir o proxecto se usa git push
  - git push origin master
  - orixin e o mesmo que antes
  - master e a rama que se desexa subir

Sincronizando

- Unha tarefa común e sincronizar os datos de remoto cos locais
- Ou sexa:
  - Ti estas a modificar o programa en local
  - Outra xente xa modificou os datos remotos
  - Para subir as túas modificacións primeiro deberás partir do mesmo punto (a versión actual)
  - Faise un pull

- Cun pull primeiro te sincronizas co servidor e logo se aplican os cambios feitos en local



- Ex: git pull origin master

Etiquetas

- Git permite agregar etiquetas en determinados momentos da historia do proxecto
- Isto tipicamente se usa para marcar os puntos de distribución. Ex: v1.0
- Iso é o que se chaman etiquetas
- As etiquetas dispoñibles se listan con:
  - git tags
- Se se busca algo concreto se pode usar:
  - git tag -l “v5.0”

- Git ten dous tipos de etiquetas:
  - Lixeiras
    - É coma un alias dun commit especifico
  - Anotadas
    - Son un obxecto git completo
    - Levan checksum
    - Garda o nome do que pon a etiqueta
    - Garda unha mensaxe
    - Se poden firmar
  - O normal é crear etiquetas anotadas

- Ex:
  - git tag -a v1.4 -m "A miña versión 1.4"
    - O -a é para que sexa anotada
    - O seguinte é a etiqueta
    - Con -m se engade a mensaxe. Se non se pon nos aparece un editor
- Se pode comprobar o contido con:
  - git show v1.4
    - Onde o v1.4 é o número de versión

- Se poden crear as etiquetas a posteriori
  - É dicir, xa fixemos commits e non lle puxemos ningunha clase de tag
- Con git log se pode obter o listado de commit
  - git log –pretty=oneline
- Unha vez que se coñece o id do commit
  - Git tag -a v1.2 id__do_teu_commit

- En remoto por defecto non se transfiren as etiquetas
  - git push origin nome_da_etiqueta
- Se hai moitas:
  - git push origin --tags