Git en servidor

GIT

- **Servidores GIT**
  - El método más recomendable para colaborar con otros es preparar un repositorio intermedio donde se pueda enviar (push) y recibir (pull) a o desde allí.
  - Nos referiremos a este repositorio como "servidor Git“
  - Solo se necesitan unos pocos recursos para albergar un repositorio Git, y, por tanto, no será necesario utilizar todo un servidor entero para él.
  - Lo primero, es elegir el/los protocolo/s que deseamos para comunicarnos con el servidor.
  - Un repositorio remoto es normalmente un repositorio básico mínimo, un repositorio Git sin carpeta de trabajo.
  - Debido a que dicho repositorio se va a utilizar exclusivamente como un punto de colaboración, no tiene sentido el tener una instantánea de trabajo (snapshot) activa en el disco (checkout); nos basta con tener solamente los propios datos Git. Básicamente, un repositorio básico mínimo son los contenidos de la carpeta .git, tal cual, sin nada más.

GIT

- **Protocolos**
  - Git puede usar cuatro protocolos principales para transferir datos:
    - Local
    - Secure Shell (SSH)
    - Git
    - HTTP.
  - Merece destacar que, con la excepción del protocolo HTTP, todos los demás protocolos requieren que Git esté instalado y operativo en el servidor.

GIT

- **Protocolo Local**
  - El más básico es el Protocolo Local, donde el repositorio remoto es simplemente otra carpeta en el disco.
  - Se utiliza habitualmente cuando todos los miembros del equipo tienen acceso a un mismo sistema de archivos, como por ejemplo un punto de montaje NFS, o en los casos en que todos se conectan al mismo ordenador.
  - Aunque este último caso no es precisamente el ideal, ya que todas las instancias del repositorio estarían en la misma máquina; aumentando las posibilidades de una pérdida.

GIT

- **Protocolo Local**
  - Si existe un sistema de archivos compartido, se puede clonar (clone), enviar (push) y recibir (pull) a/desde repositorios locales basado en archivos.
  - Para clonar un repositorio, o para añadirlo como remoto a un proyecto ya existente, se usa la ruta (path) del repositorio como su URL.
  - Por ejemplo, para clonar un repositorio local, puedes usar algo como:

```
$ git clone /opt/git/project.git
O
$ git clone file:///opt/git/project.git
```

  - Para añadir un repositorio local a un proyecto Git existente:

```
git remote add local_proj /opt/git/project.git
```

GIT

- **Procotolo SSH**
  - SSH es el protocolo más habitual para Git.
  - SSH es el único protocolo de red con el que se puede fácilmente tanto leer como escribir.
  - Los otros dos protocolos de red (HTTP y Git) suelen ser normalmente protocolos de solo-lectura; de tal forma que, aunque estén disponibles para el público en general, se sigues necesitando SSH para escritura.
  - Otra ventaja de SSH es el su mecanismo de autentificación, sencillo de habilitar y de usar.

GIT

- **Procotolo SSH**
  - Para clonar un repositorio a través de SSH, se puede indicar una URL ssh://

```
$ git clone ssh://user@server:project.git
```

  - Se puede prescindir del protocolo. Git asume SSH si no se indicas nada expresamente:

```
$ git clone user@server:project.git
```

GIT

- **Protocolo Git**
  - El protocolo Git es un demonio (daemon) especial, que viene incorporado con Git.
  - Escucha por un puerto dedicado (9418), y da un servicio similar al del protocolo SSH, pero sin ningún tipo de autentificación.
  - Para que un repositorio pueda exponerse a través del protocolo Git, hay que crear un archivo git-export-daemon-ok. Sin este archivo, el demonio no hará disponible el repositorio.
  - Pero, aparte de esto, no hay ninguna otra medida de seguridad. O el repositorio está disponible para que cualquiera lo pueda clonar, o no lo está.
  - Lo cual significa que, normalmente, no se podrá enviar (push) a través de este protocolo.
  - Aunque realmente si que puedes habilitar el envio, dada la total falta de ningún mecanismo de autentificación, cualquiera que encuentre la URL a tu proyecto en Internet, podrá enviar (push) contenidos a él.
  - Es decir hay que huir de este protocolo si queremos escribir

GIT

- **Protocolo HTTP/S**
  - Por último, tenemos el protocolo HTTP,
  - Es fácil habilitarlo.
  - Basta con situar el repositorio Git bajo la raiz de los documentos HTTP y preparar el enganche (hook) post-update adecuado.
  - A partir de ese momento, cualquiera con acceso al servidor web podrá clonar tu repositorio.

```
$ cd /var/www/htdocs/
$ git clone --bare /path/to/git_project gitproject.git
$ cd gitproject.git
$ mv hooks/post-update.sample hooks/post-update
$ chmod a+x hooks/post-update
```

GIT

- **Protocolo HTTP/S**
  - El enganche post-update que viene de serie con Git se encarga de lanzar el comando adecuado (git update-server-info) para hacer funcionar la recuperación (fetching) y el clonado (cloning) vía HTTP.
  - Este comando se lanza automáticamente cuando envias (push) a este repositorio vía SSh, de tal forma que otras personas puedan clonarlo:

```
$ git clone http://example.com/gitproject.git
```

GIT

- **Montar un servidor**
  - El primer paso para preparar un servidor Git, es exportar un repositorio existente a un nuevo repositorio básico, a un repositorio sin carpeta de trabajo.
  - Se usa el comando clone con la opción --bare.
  - Por convenio, los nombres de los repositorios básicos suelen terminar en .git.

```
$ git clone --bare my_project my_project.git
Initialized empty Git repository in /opt/projects/my_project.git/
```

GIT

- **Poner el repositorio básico en un servidor**
  - Ahora solo hay que copiarlo en un servidor y configurar los protocolos.
  - Por ejemplo, si tenemos un servidor con ssh podemos copiar el repositorio:

```
$ scp -r my_project.git user@git.example.com:/opt/git
```

  - De ese modo, cualquier otro usuario con acceso de lectura SSH a la carpeta /opt/git del servidor, podrá clonar el repositorio con el comando:

```
$ git clone user@git.example.com:/opt/git/my_project.git
```