**Git on the server**

A remote repository is generally a ***bare repository*** — a Git repository that has no working directory. Because the repository is only used as a collaboration point, there is no reason to have a snapshot checked out on disk; it’s just the Git data. In the simplest terms, a bare repository is the contents of your project’s .git directory and nothing else.

**$git init --bare**

Git can use four distinct protocols to transfer data: Local, HTTP, Secure Shell (SSH) and Git.

**Local Protocol**

* The most basic is the *Local protocol*, in which the remote repository is in another directory on the same host. This is often used if everyone on your team has access to a shared filesystem such as an NFS mount, or in the less likely case that everyone logs in to the same computer.
* The latter wouldn’t be ideal, because all your code repository instances would reside on the same computer, making a catastrophic loss much more likely.
* If you have a shared mounted filesystem, then you can clone, push to, and pull from a local filebased repository.
* To clone a repository like this, or to add one as a remote to an existing project, use the path to the repository as the URL
* $ git clone <path> >>> /srv/git/project.git or <file:///srv/git/project.git>
* If you just specify the path, Git tries to use hardlinks or directly copy the files it needs.
* If you specify file://, Git fires up the processes that it normally uses to transfer data over a network, which is generally much less efficient. The main reason to specify the file:// prefix is if you want a clean copy of the repository.
* **The Cons**
* The cons of this method are that shared access is generally more difficult to set up and reach from multiple locations than basic network access. If you want to push from your laptop when you’re at home, you have to mount the remote disk, which can be difficult and slow compared to network based access.
* A repository on NFS is often slower than the repository over SSH on the same server, allowing Git to run off local disks on each system.

Finally, this protocol does not protect the repository against accidental damage. Every user has full shell access to the “remote” directory, and there is nothing preventing them from changing or removing internal Git files and corrupting the repository.

**Smart HTTP & HTTPS Protocol**

$ git clone https://example.com/gitproject.git

**The SSH Protocol**

* $ git clone ssh://[user@]server/project.git
* $ git clone [user@]server:project.git
* **The Cons**
* The negative aspect of SSH is that it doesn’t support anonymous access to your Git repository. If you’re using SSH, people *must* have SSH access to your machine, even in a read-only capacity, which doesn’t make SSH conducive to open source projects for which people might simply want to clone your repository to examine it.

**Git protocol**

* it listens on a dedicated port (**9418**) that provides a service similar to the SSH protocol, but with absolutely no authentication.
* In order for a repository to be served over the Git protocol, you must create a
* git-daemon-export-ok file — the daemon won’t serve a repository without that file in it — but, other than that, there is no security.

**Setting up git repo on own server**

create a single *git* user account on the machine, ask every user who is to

have write access to send you an SSH public key, and add that key to the ~/.ssh/authorized\_keys file of that new *git* account. At that point, everyone will be able to access that machine via the *git*

account.

$ sudo adduser git

$ su git

$ cd

$ mkdir .ssh && chmod 700 .ssh

$ touch .ssh/authorized\_keys && chmod 600 .ssh/authorized\_keys

Next, you need to add some developer SSH public keys to the authorized\_keys file for the git user. Now, you can set up an empty repository for them by running git init with the --bare option, which initializes the repository without a working directory:

$ cd /srv/git

$ mkdir project.git

$ cd project.git

$ git init --bare

Initialized empty Git repository in /srv/git/project.git/