3. Important terminology in Git

3.1. Cloning, creating and deleting a Git repository

The process of copying an existing Git repository is called cloning. After cloning a repository the user has the complete repository with its history on his local machine. Of course, Git also supports the creation of new repositories.

If you want to delete a Git repository, you can simply delete the folder which contains the repository.

3.2. Bare repositories and non-bare repositories

If you clone a Git repository, by default, Git assumes that you want to work in this repository as a user. Git also supports the creation of repositories targeting the usage on a server.

- bare repositories are supposed to be used on a server for sharing changes coming from different developers. Such repositories do not allow the user to modify locally files and to create new versions for the repository based on these modifications.
- non-bare repositories target the user. They allow you to create new changes through modification of files and to create new versions in the repository. This is the default type which is created if you do not specify any parameter during the clone operation.

A local non-bare Git repository is typically called *local repository*.

3.3. Working tree

A local repository provides at least one collection of files which originate from a certain version of the repository. This collection of files is called the *working tree*. It corresponds to a checkout of one version of the repository with potential changes done by the user.

The user can change the files in the *working tree* by modifying existing files and by creating and removing files. Afterwards he can add these changes to the repository.

3.4. Local operations

Once the user has his local repository, he can perform modify files in his working tree and perform version control operations. For example he can create new versions for the files in his Git repository, revert the files to another version stored in the repository, etc.

3.5. Synchronization with remote repositories

Git allows the user to synchronize the local repository with other (remote) repositories.

Users with sufficient authorization can send new version in their local repository to to remote repositories via the *push* operation. They can also integrate changes from other repositories into their local repository via the *fetch* and *pull* operation.

3.6. The concept of branches

Git supports *branching* which means that you can work on different versions of your collection of files. A branch separates these different versions and allows the user to switch between these versions to work on them.

For example, if you want to develop a new feature, you can create a branch and make the changes in this branch without affecting the state of your files in another branch.

Branches in Git are local to the repository. A branch created in a local repository, which was cloned from another repository, does not need to have a counterpart in the remote repository. Local branches can be compared with other local branches and with *remote-tracking branches*. A remote-tracking branch proxies the state of a branch in another remote repository.

Git supports the combination of changes from different branches. This allows the developer, for example, to work independently on a branch called *production* for bugfixes and another branch called *feature_123* for implementing a new feature. The developer can use Git commands to combine the changes at a later point in time.

For example, the Linux kernel community used to share code corrections (patches) via mailing lists to combine changes coming from different developers. Git is a system which allows developers to automate such a process.