Branches

Branches can be used to store independent series of commits in the repository, e.g., to fix bugs for a released software project while simultaneously developing new features for the next project version.

Git distinguishes between two kinds of branches: *local branches* and *remote branches*. In the local repository, you can create as many local branches as you like. Remote branches, on the other hand, are local branches of the origin repository. In other words: Cloning a remote repository clones all its local branches which are then stored in your local repository as remote branches. You can't work directly on remote branches, but have to create local branches, which are 'linked' to the remote branches. The local branch is called *tracking branch*, and the corresponding remote branch *tracked branch*. Local branches can be tracking branches, but they don't have to.

The default local main branch created by Git is named *master*, which is analogous to SVN's *trunk*. When cloning a remote repository, the master tracks the remote branch *origin/master*.

Working with Branches

When you push changes from your local branch to the origin repository, these changes will be propagated to the tracked (remote) branch as well. Similarly, when you pull changes from the origin repository, these changes will also be stored in the tracked (remote) branch of the local repository. To get the tracked branch changes into your local branch, the remote changes have to be *merged from the tracked branch*. This can be done either directly when invoking the *Pull* command in SmartGit, or later by explicitly invoking the *Merge* command. An alternative to the Merge command is the Rebase command.

> **Tip**
>
> The method to be used by Pull (either *Merge* or *Rebase*) can be configured in **Repository|Settings** on the **Pull** tab.

Branches are just pointers to commits

Every branch is simply a named pointer to a commit. A special unique pointer for every repository is the *HEAD* which points to the commit the working tree state currently corresponds to. The HEAD cannot only point to a commit, but also to a local branch, which itself points to a commit. Committing changes will create a new commit on top of the commit or local branch the HEAD is pointing to. If the HEAD points to a local branch, the branch pointer will be moved forward to the new commit; thus the HEAD will also indirectly point to the new commit. If the HEAD points to a commit, the HEAD itself is moved forward to the new commit.