Branching guidelines

Last updated by | Nils Johannsen | May 23, 2024 at 12:50 PM GMT+2

We use git as version control system of our source code. Git allows very cheap usage of branches and extensive possibilities for merges.

The *main* branch

The default and protected branch is called *main*. In legacy repositories it is also called *master*. This branch is protected by policies whose require any changes by pull requests with at least one reviewer to approve the changes. Only the initial push to a repository allows to commit to this branch without review. The *main* branch has to be maintained carefully and should always be the single major, stable and releasable branch.

The *version* branches

In case you intend diverging *main* branches, because of different variants of your sample, add a category to your *main* branch separated by a slash. This is very likely if your sample has diverged by one variant for TwinCAT 4024 and a second variant for TwinCAT 4026. In that case the following branch names may apply:

- `main` for the latest version
- `main/4024` for TwinCAT 4024
- `main/4026` for TwinCAT 4026

The *feature* branches

If you start to change anything in the *main* branch, start with creating a new *feature* branch. Common naming of such *feature* branches starting with your domain account user name and a short description, separated by a slash to establish categories and using minus instead of spaces. In case you are woking on a feature with potentially multiple branches, the extension of a version of the branch is recommended. As branches are cheap in git and you are encouraged to use many of them, but name them carefully to avoid confusion by others and yourself.

- `<user>/<feature>`
- `NilsJ/add-new-features`
- `NilsJ/feature-a-v1` , `NilsJ/feature-a-v2`
- `NilsJ/feature-a/v1` , `NilsJ/feature-a/v2`