SDEVEN Software Development & Engineering Methodology

Version: 7.0.5
Release date: 230620

Branches (SDEVEN.40-BRAN)

Table of Content

Preamble

This section refers the current policies regarding *git repository* branches.

Classification

The branches are classified like that:

- *mandatory* (always required). As example is the **master** (or *main*) branch. The name *master* is preferred (as it is already used in automation scripts), but sometimes the name main can be found (as with best practices recommendations starting with spring of 2021 year).
- with a *long life cycle* (aka just *development*). In this category is **development** branch which is required up to a minor release, then the name can be changed if needed to reflect versioning policy. Another example could be *release* beach intended for documentation review and finalizing and packaging of system.
- *test* branches which are created usually from an *alpha* or *beta* tag and kept until the required tests are passed. This kind of branches can return (merge) back to *development* corresponding branch or to a **release** branch if tests passed for preparing a potential release. In both cases, after test finalizing and branch merge, it will be deleted.
- *developer dedicated* branches local kept on remote git for a quick reference for all developers without requiring a git client
- *personal* branches reflecting current work for developers that: (i) have a `git` client installed or (ii) remote for developers using mobile devices without a `git` client installed. For such as branches open on public remote[1] `git`, the name of developer and phrase "`dev`" or "`phone`" and these branches *need to be requested from DevOps person* in order to crete them, give them enough rights and not to be (automatically) dropped.

Branches used and their names

Usually, as not stated otherwise in a project, the following branches should be used:

- **development** - consolidates development of all team. Branch is *permanent* and set as *default*
- **master** - current version of system in production. Branch is *permanent*
- **xxx-dev** - work branch for team member `xxx`. Branch is *temporary* and should be administered by the DevOps person. This kind of branch is made to work from different devices where a `git` client cannot be used and files must be individually uploaded or edit using the `git` system web interface. The person for whom it was created can receive full rights on this branch
- **qa_test** / test - *used according to classification. Branch is* temporary*
- **release** / `version_string`**-rel** - used according to classification. Branch is *temporary*

master branch name

starting from 2022 new git systems are default configured to name default base branch as `main` instead of `master`. Please check to avoid mistakes due to branch name *use in different automation / configuration* `yaml` scripts.

Frequent mistakes

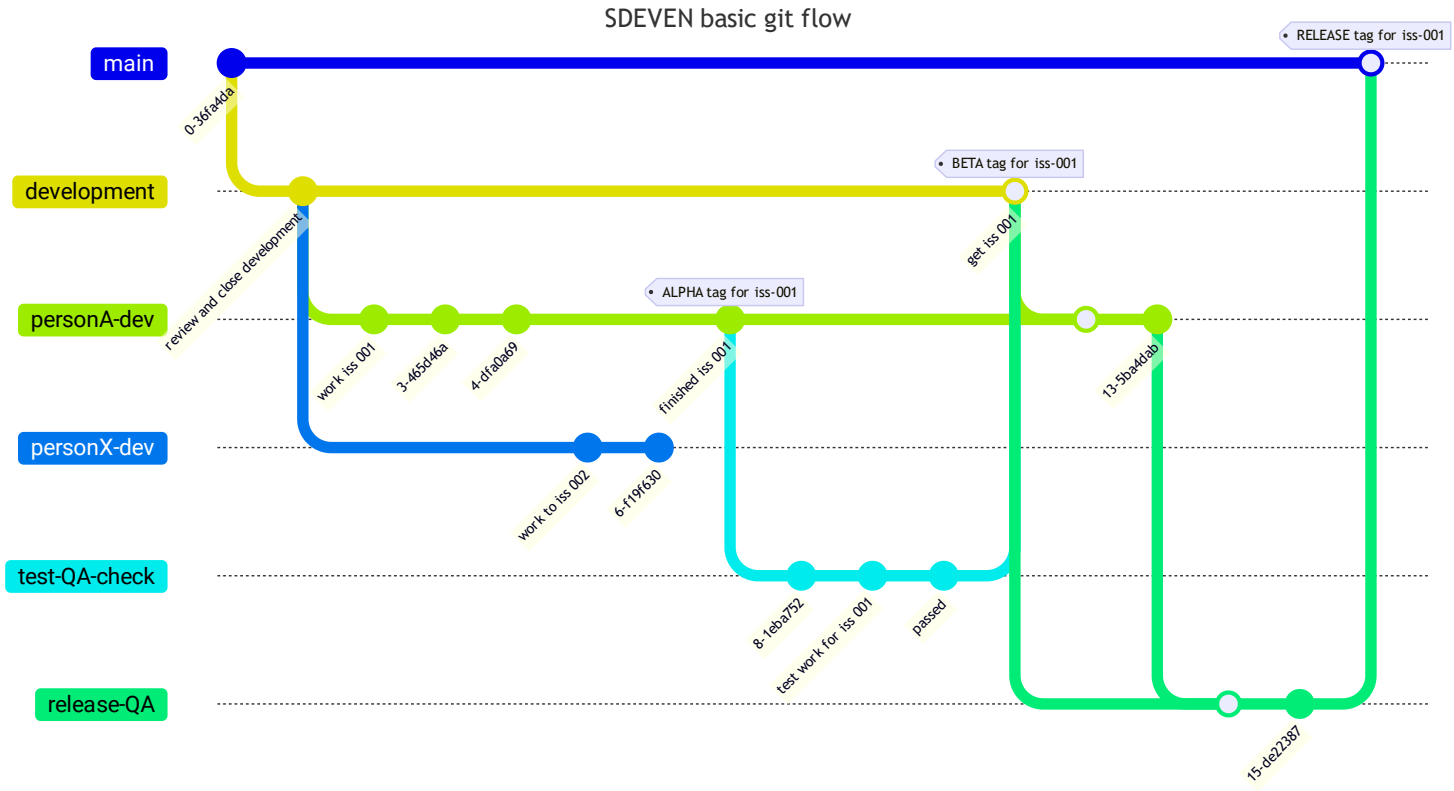
- names containing "`dev`" are used to signify a "local" developer work. These should not be confused with `development` branch which has already an explained purpose. They are used by people that need a repository for their current work and cannot install a local client.
- names containing "`rel`" or "`tst`" or "`qa`" are intended for test purposes. This situation could appear (i) for own tests, (ii) for tests done by other people or (iii) for tests made before a release. Also, instead of making "release or test branches" when intention is for example not commit work but have a kind of snapshot with *"AS IS NOW"*, a tag can be created and transformed latter in a branch. This method is *STRONGLY RECOMMENDED* for any kind of snapshots that are needed. Just pay attention with these tags and DO NOT EXPECT a long life cycle for them as anytime can be dropped by a `DevOps` that remarks they are out of used conventions and policies - a better option is just to communicate the intention.

Tagging recommendations

- for consistency, long term and reference tags should be named using *RENVER* conventions
- working tags should contain words like `dev` or `adev` *(from alpha dev)* if not an *alpha* release is intended. This will help for a right alphabetically sort, normally their preceding an *alpha* or upper release.

Graphic basic flow

This diagram shows the basic flow for `master (main)`, `development` and one `xxx-dev` branch. Also on graphic the practices ref tagging are shown:



Names and codes used in diagram:

- `iss-001` is an issue that needs to be closed (and coded)
- `personA-dev` is a *personal* branch created for "person A" to work (in example for issue 001)
- `personX-dev` is a *personal* branch created for another developer (persona X)

Also diagram shows different tags created as occasioned by "issue 001".

1. the internal `git` used by team ↩

Last update: July 18, 2023