Git Workflow Changes and Tricks

Dave van Soest

April 25, 2014

Workflow changes

Responsibilities

Tricks

Questions & Discussion

Workflow changes

- Merge branches to master without squashing.

Workflow changes

- Merge branches to master without squashing.
  - Yes, you can have multiple commits in a branch...

Workflow changes

- Merge branches to master without squashing.
  - Yes, you can have multiple commits in a branch...
- Don't delete feature/story branches.

Workflow changes

- Merge branches to master without squashing.
  - Yes, you can have multiple commits in a branch...
- Don't delete feature/story branches.
  - After merging the branch is basically a tag on the master branch.

Responsibilities

Developers

Responsibilities

Developers

- Deliver branches with only self-contained commits, which are logical units of work.

Responsibilities

Developers

- Deliver branches with only self-contained commits, which are logical units of work.
- Prefer combined commits over separate commits.

Responsibilities

Developers

- Deliver branches with only self-contained commits, which are logical units of work.
- Prefer combined commits over separate commits.
- During review, check commits to be self-contained and whether they are combined as much as logically possible.

Responsibilities

Developers

- Deliver branches with only self-contained commits, which are logical units of work.
- Prefer combined commits over separate commits.
- During review, check commits to be self-contained and whether they are combined as much as logically possible.
- During review, check commit message quality.

Responsibilities

Developers

- Deliver branches with only self-contained commits, which are logical units of work.
- Prefer combined commits over separate commits.
- During review, check commits to be self-contained and whether they are combined as much as logically possible.
- During review, check commit message quality.

Git master

Responsibilities

Developers

- Deliver branches with only self-contained commits, which are logical units of work.
- Prefer combined commits over separate commits.
- During review, check commits to be self-contained and whether they are combined as much as logically possible.
- During review, check commit message quality.

Git master

- Before merging to master, check commits to be self-contained and whether they are combined as much as logically possible.

Responsibilities

Developers

- Deliver branches with only self-contained commits, which are logical units of work.
- Prefer combined commits over separate commits.
- During review, check commits to be self-contained and whether they are combined as much as logically possible.
- During review, check commit message quality.

Git master

- Before merging to master, check commits to be self-contained and whether they are combined as much as logically possible.
- Before merging to master, check commit message quality.

Separate commits

Question

When to combine commits?

Separate commits

Question

When to combine commits?

Answer

When not to combine commits?

Separate commits

Question

When not to combine commits?

Separate commits

Question

When not to combine commits?

Answer

Having multiple commits should be the exception. It can be warranted in the following cases (non-exhaustive):

Separate commits

Question

When not to combine commits?

Answer

Having multiple commits should be the exception. It can be warranted in the following cases (non-exhaustive):

- Very big stories, having multiple logically separate technical features.

Separate commits

Question

When not to combine commits?

Answer

Having multiple commits should be the exception. It can be warranted in the following cases (non-exhaustive):

- Very big stories, having multiple logically separate technical features.
- Stories having multiple completely independent sub-stories.

Separate commits

Question

When not to combine commits?

Answer

Having multiple commits should be the exception. It can be warranted in the following cases (non-exhaustive):

- Very big stories, having multiple logically separate technical features.
- Stories having multiple completely independent sub-stories.
- Out-of-scope code quality improvements.

Commit messages

Commit messages should be...

- written in clear and correct English;

Commit messages

Commit messages should be...

- written in clear and correct English;
- cover every change made in the commit;

Commit messages

Commit messages should be...

- written in clear and correct English;
- cover every change made in the commit;
- as short as possible (while not violating the above);

Commit messages

Commit messages should be...

- written in clear and correct English;
- cover every change made in the commit;
- as short as possible (while not violating the above);
- 'tagged' if they contain out-of-scope changes;

Commit messages

Commit messages should be...

- written in clear and correct English;
- cover every change made in the commit;
- as short as possible (while not violating the above);
- 'tagged' if they contain out-of-scope changes;
- conforming to all other rules set up by the team.

Tricks

- Stashing

Tricks

- Stashing
- Staging

Tricks

- Stashing
- Staging
- Diffing

Tricks

- Stashing
- Staging
- Diffing
- Amending commits

Tricks

- Stashing
- Staging
- Diffing
- Amending commits
- Interactive rebasing

Tricks

- Stashing
- Staging
- Diffing
- Amending commits
- Interactive rebasing
- Be careful with force pushing

Tricks

- Stashing
- Staging
- Diffing
- Amending commits
- Interactive rebasing
- Be careful with force pushing
- Working with multiple developers on a feature

Questions & Discussion