Barcelona, 17 May 2016

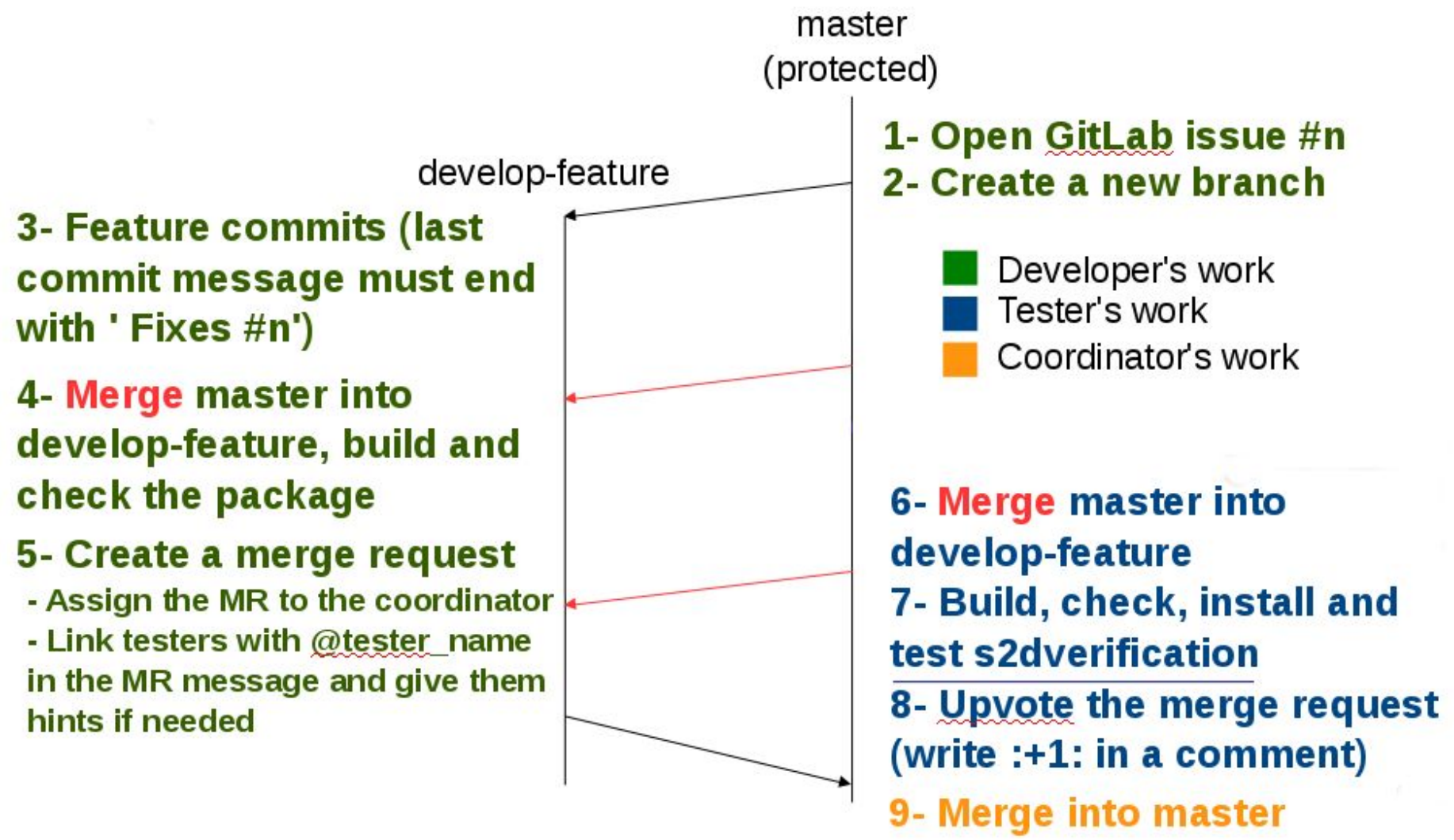
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Git branching strategies

Support software

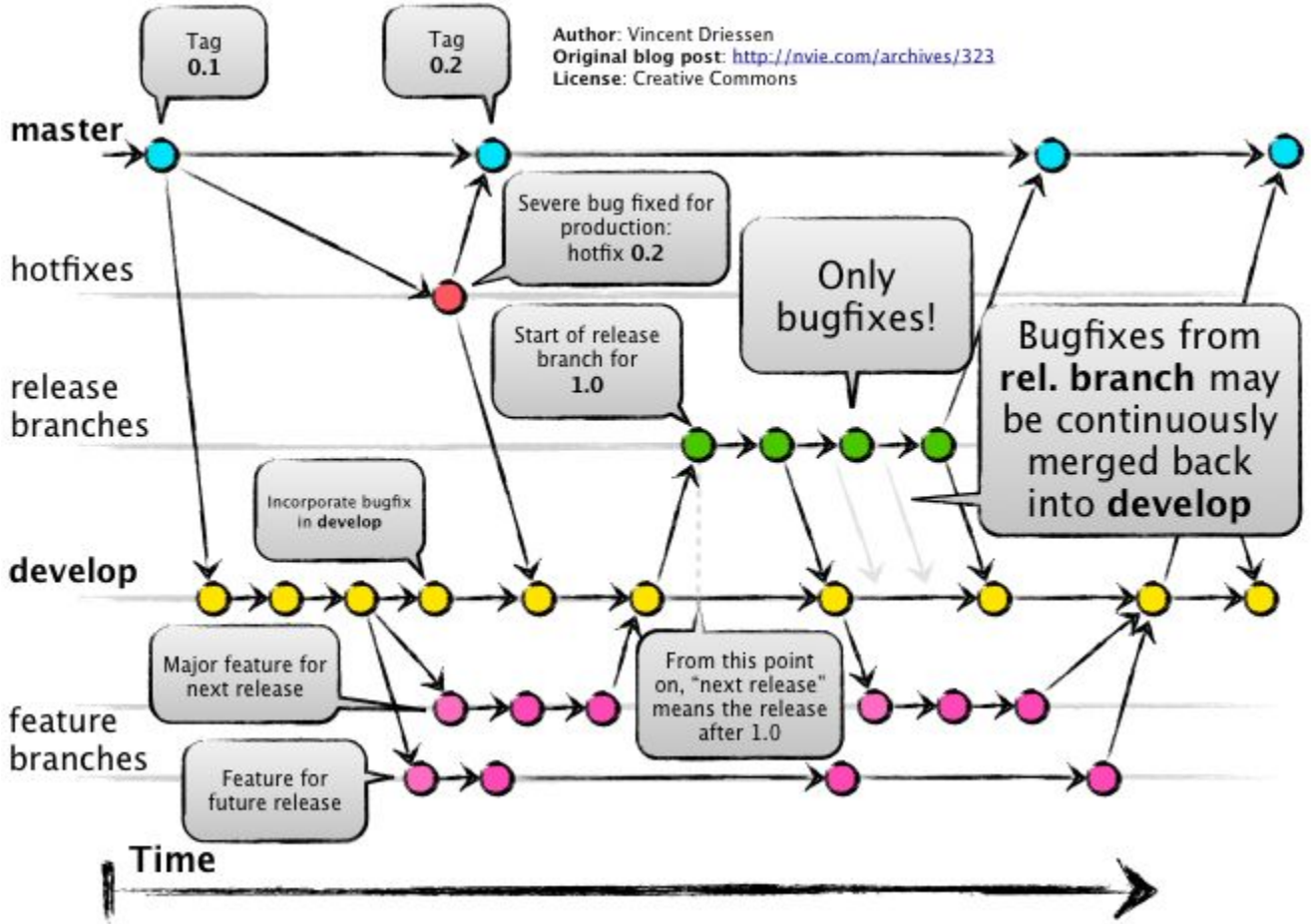
Jordi Cuadrado, Joan López, Domingo Manubens, Nicolau Manubens

GitHub flow



Recap

Git flow



Recap

- Two possible strategies...
  - **GitHub-flow**
  - **Git-flow**

GitHub-flow	Git-flow
Unstable versions in the master branch	Master always contains stable versions
Uncontrolled time window when merg ing tested+accepted feature to master	Full control of changes that are released from develop to master
Simple	Extra complexity (for CES & scientists)
GitLab issues close after merge to master but before release: oriented to developers	GitLab issues close only after release: oriented to users
Default target branch in GitLab merge requests: master	Default target branch in GitLab merge requests: master
Too simple for big projects	Scales for large teams
Appropriate for fast-paced developments	Too heavy for fast-paced developments (GitFlow commands can help).

News

- Third strategy...
  - **GitLab-flow**
- Developments happen in feature branches that branch off master branch, pretty much as in GitHub-flow.
- There are production branches to make clear which versions are stable.
- CI tests are run on master.
- This strategy is in the middle of the other two: it allows to use only master and features branches in simple projects, or it also allows to do more serious developments with production branches that contain the stable releases.

https://about.gitlab.com/2014/09/29/gitlab-flow/
http://docs.gitlab.com/ee/workflow/gitlab_flow.html

- Which should we choose? Let's see some support tools.

Support tools

- GitFlow
  - Terminal commands that make easy typical branching or merge operations in a Git-flow branching model.
  - Helpful for developers (CES) not for scientists, they can't merge or release.

- GitLab plug-ins:
  - Only to trigger testing in production branches.

- GUIs:
  - SmartGit (all popular SOs):
    - https://blogs.endjin.com/2015/01/using-smartgit-to-follow-the-gitflow-branching-and-workflow-model/
  - SourceTree (only Windows & Mac):
    - https://www.sourcetreeapp.com/
  - GitTower (only 30-day trial):
    - https://www.git-tower.com/learn/git/ebook/en/mac/advanced-topics/git-flow#start
  - Others: https://atom.io/packages/git-control

SmartGit

+ Allows to manage branches, commits, pushes, … from a GUI. One can do without Git terminal commands.
+ Automatizes Git-flow and GitHub-flow strategies.
+ Compatibility with sub-modules.
+ Straightforward point&click use, integrates a more complete diff viewer, could be of help to get onboard users currently lagging at git?
- GitLab still required for user authentication, merge requests, discussions, milestones, …
- Additional tool. Adds complexity and, potentially, confusion.
- One could forget how to use Git, trouble if working remotely.

Discussion

- How to proceed now
  - Which branching strategy do we pick
    - Simplified version of Git-flow / GitHub-flow + develop branch?
    - GitHub-flow?
    - Two different strategies, and see which projects use which?
  - Which development/testing strategy do we pick
    - Do we apply the development strategy to all projects with scientists developing?
      - a) Create issue for new development, b) Do the development, c) Open a merge request and assign 2 testers, d) Testers to review and approve, e) Coordinator to review and merge.
    - Do we apply Continuous Integration in software without scientists?

      (run unit tests before merging feature, run integration tests repeatedly on develop → Jenkins?)

Discussion

- How to proceed now
  - What software stack do we suggest to the scientists
    - Do we suggest using Git/SmartGit + GitLab [+ testing scripts for the testing procedures in s2dv]?
  - What software stack do we suggest to the developers
    - Git/SmartGit + GitLab + GitFlow?
  - Should we do a presentation to scientists?
  - What should be the deadlines?

www.bsc.es

Thank you!

For further information please contact
ces@bsc.es