The git-flow model

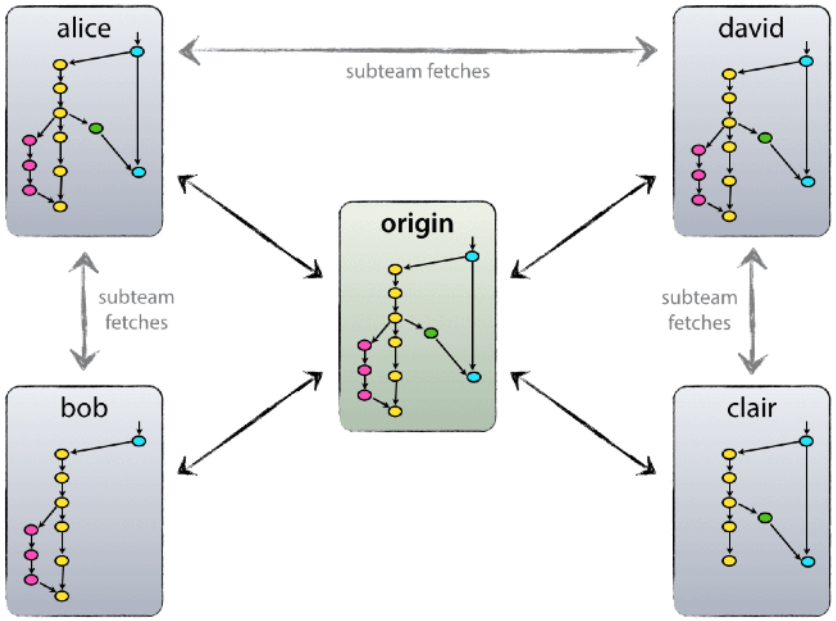
Giovanni Pizzi

Theory and Simulation of Materials, EPFL Lausanne

FNSNF MARVEL NATIONAL CENTRE OF COMPETENCE IN RESEARCH MAX DRIVING THE EXASCALE TRANSITION EPFL

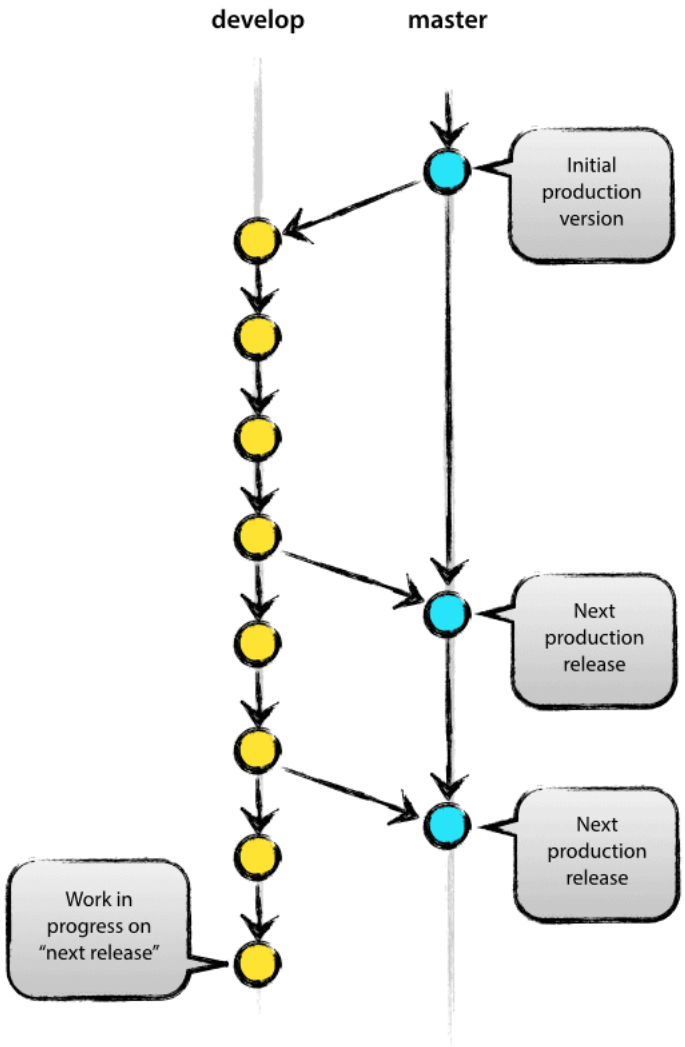
The git-flow model

- https://nvie.com/posts/a-successful-git-branching-model/ (and rediscussed elsewhere, e.g. https://datasift.github.io/gitflow/IntroducingGitFlow.html)
- Extensions. e.g.: http://tleyden.github.io/blog/2014/04/09/a-successful-git-branching-model-with-enterprise-support/
- Based on fork/ pull request model



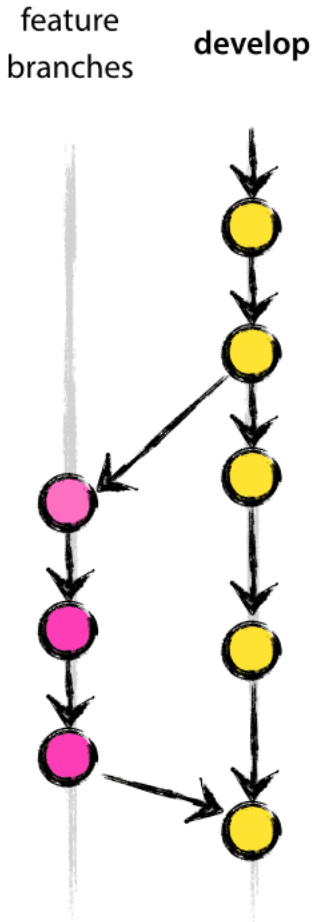
Main branches

- **master**: last stable production release
- **develop**: latest delivered development changes for the next release
- Making a commit to *master* means making a new release
- On top, tag versions with vX.Y.Z, using semantic versioning



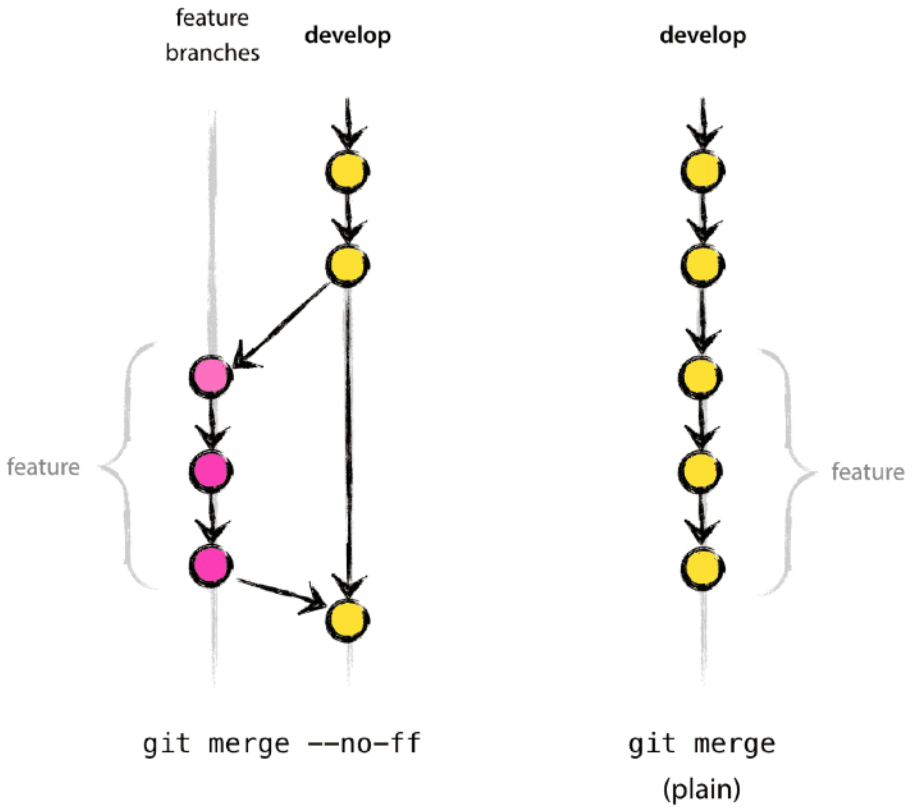
Support branches

- May branch off from: **develop**
- Must merge back into: **develop**
- Branch naming convention:
  - anything except master, develop, release-*, or hotfix-*
  - In AiiDA: we (try to) use the syntax fix-<ISSUE_NUM>-SOME-NAME
- develop new features for the upcoming or a distant future release.
- The target release in which this feature will be incorporated may well be unknown at branching-off time
- *Think well on how to deal with too long/ complex/code-breaking features*



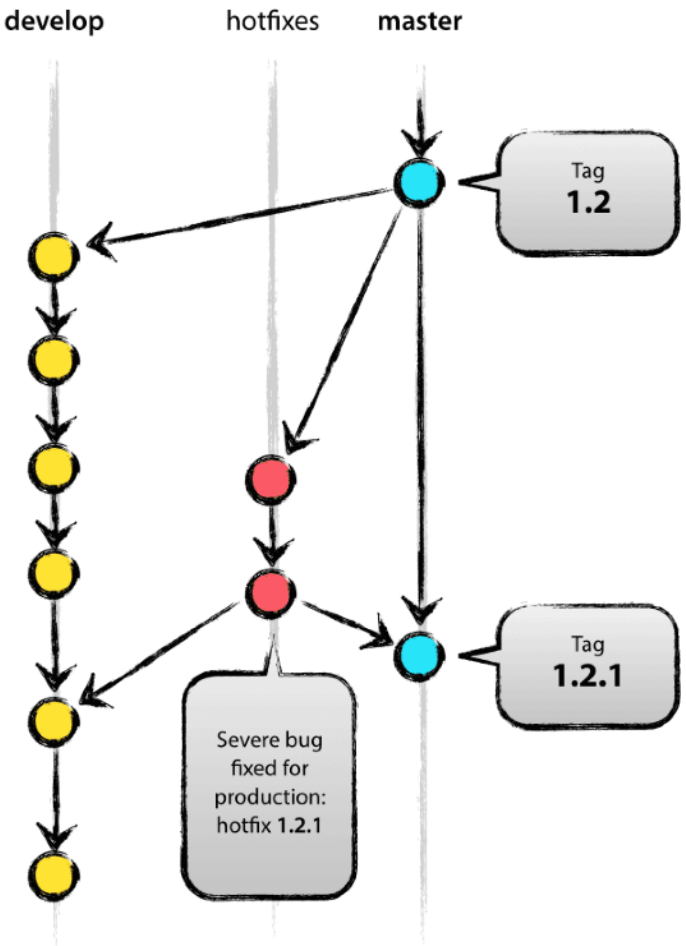
How to merge back

- Distinguish between git features (commits, merge commits) and GitHub features (issues, pull requests)
- Note the difference between merge, merge --no-ff, rebase, squash, …
- If possible:
  - Favour cleanliness of your history in git and of the content of the git metadata
  - Squash/rebase useless commits
  - Think if you want to keep a branch in the history or not
- Git-flow suggestion: use --no-ff (or, what we do: squash in a single commit if appropriate)

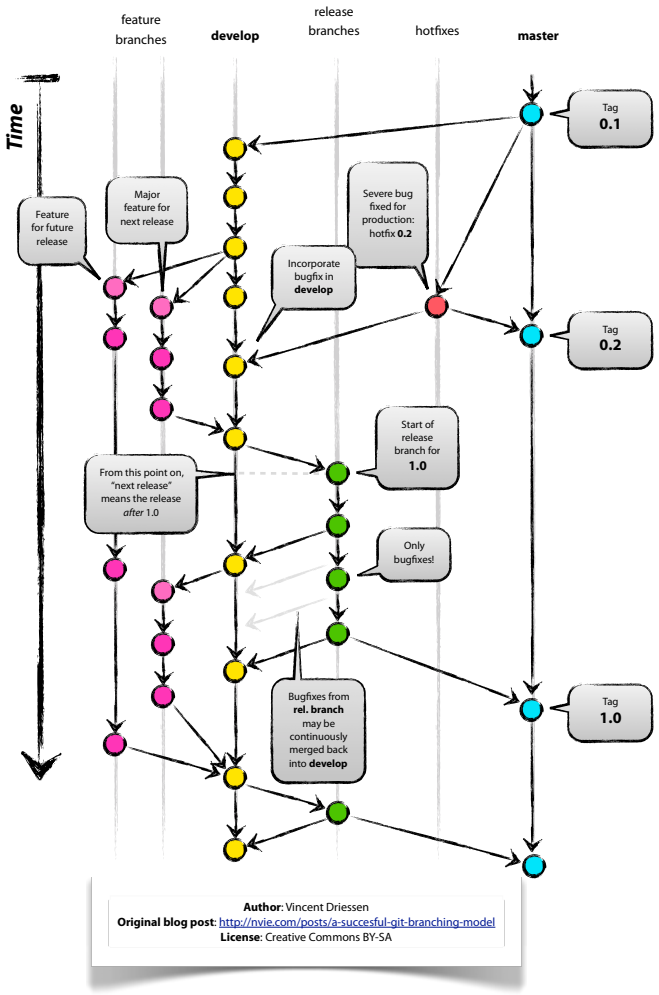


Hotfix branches

- Create a hot fix branch to fix something the latest release, without adding the new features in develop
- Then merge it back into master (to make a release) and into develop (to have the features in the most recent development branch)



The (original git-flow model)



Extension to support multiple support branches (for older releases)

- From http://tleyden.github.io/blog/2014/04/09/a-successful-git-branching-model-with-enterprise-support/; note that they call differently develop/master as master/stable…
- Keep a 'support/x.y.z' branch open, branched off the relevant release tag commit, and make tags out of it.
- "Cherry-pick" from develop or apply fixes directly as in a new branch. Do NOT merge back into develop
- It makes sense for "old" releases, where the codebase changed significantly (and you want to support older releases for bugfixes)