1. What is Git, write atleast 5 points?

Git is free and open-source software distributed under GNU General Public License Version 2.

Git is a version control / version tracking system.

It keeps tracking changes in source code during software development i.e. it is a source code management tool.

It is designed for coordinating work among programmers, and can be used to track changes in any set of files.

It is open source distributed system and does not have the concept that only centralised server will contain the original source code.

In short, git stores the entire history of a software project in a repository, and introduce changes to that repository in a controlled way, allowing coordination between many people without accidentally overwriting or corrupting files.

2. What is the advantage & disadvantage of Git?

Advantages of Git:

Staging Area: Git has an intermediate stage called “index” or “staging area” where commits can be formatted and modified before completing the commit.

Security: Git handles your security with cryptographic method SHA-1. The algorithm manages your versions, files and directory securely so that your work is not corrupted.

Distributed: Git is distributed in nature. Distributed means that the repository or the complete code base is mirrored onto the developer’s system so that he can work on it only.

Open source: open source invites the developers from all over the world to contribute to the software and make it more and more powerful through features and additional plugins.

Performance: Committing, branching, merging all are optimized for a better performance than other systems.

Branching Model: Git has a different branching model than the other VCS. Git branching model lets you have multiple local branches which are independent of each other. Having this also enables you to have friction-less context switching (switch back and forth to new commit, code and back), role-based code (a branch that always goes to production, another to testing etc) and disposable experimentation (try something out, if does not work, delete it without any loss of code).

Disadvantages of Git:

GIT requires technical excellence and it is slower on windows. They have tedious command lines to input and doesn’t track renames.

They have poor GUI and usability. And also, they take a lot of resources which slows down the performance.

GIT doesn’t support checking out sub-trees. For each project, the central service would need to be set up for multiple package repositories.

It lacks window support and doesn’t track empty folders.

GIT needs multiple branches to support parallel developments used by the developers.

There is no built-in access control and doesn’t support binary files.

They do not provide access control mechanisms in case of security.

The process of Packing is very expensive completely.

3. Git is written in which language?

Git is mainly written in C, Bourne Shell to provide wrappers around it., Perl for some tools and a little bit of C++, Python, Tcl.

4. How will you create a repository in git?

Using git init inside the directory to be used as repository, then add required files and add them for staging and commit using git add . And git commit -m “<message>”

5. What does “git push” command do in Git?

git push command pushes the files from the present branch in the local repository into the branch of remote repository which is specified as input / set to sync with the present branch in the local repository.

6. What is a commit message in Git?

Commit message is a comment on what has been committed which will be helpful for collaborators.

7. What's the difference between a "pull request" and a “branch"?

A branch is just a separate version of the code. A pull request is when someone take the repo, makes their own branch, does some changes, then tries to merge that branch in (put their changes in the other person's code repository).

8. What is the difference between "git pull" and "git fetch”?

git fetch is the command that tells your local git to retrieve the latest meta-data info from the original (yet doesn't do any file transferring. It's more like just checking to see if there are any changes available). git pull on the other hand does that AND brings (copy) those changes from the remote repository into local repository.

9. How to revert previous commit in git?

Use git revert <SHA-id>

10. How to remove a file from git without removing it from your file system?

Use git rm --cached filename1 filename2 to remove the files from git without deleting them from file system

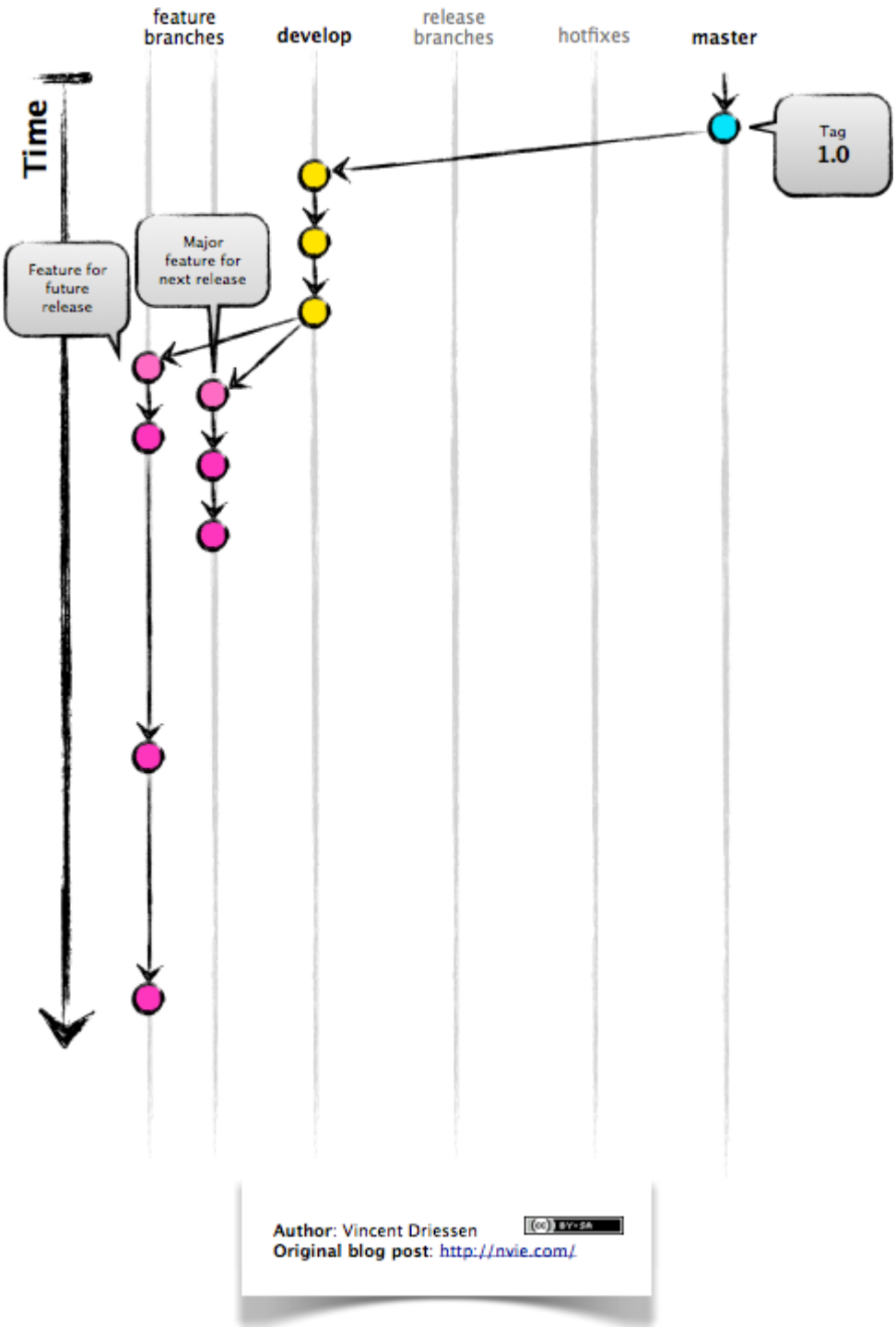
And then commit the changes after staging. ‘-rf’ can be used for forcefully deleting them

11. When do you use "git rebase" instead of "git merge”?

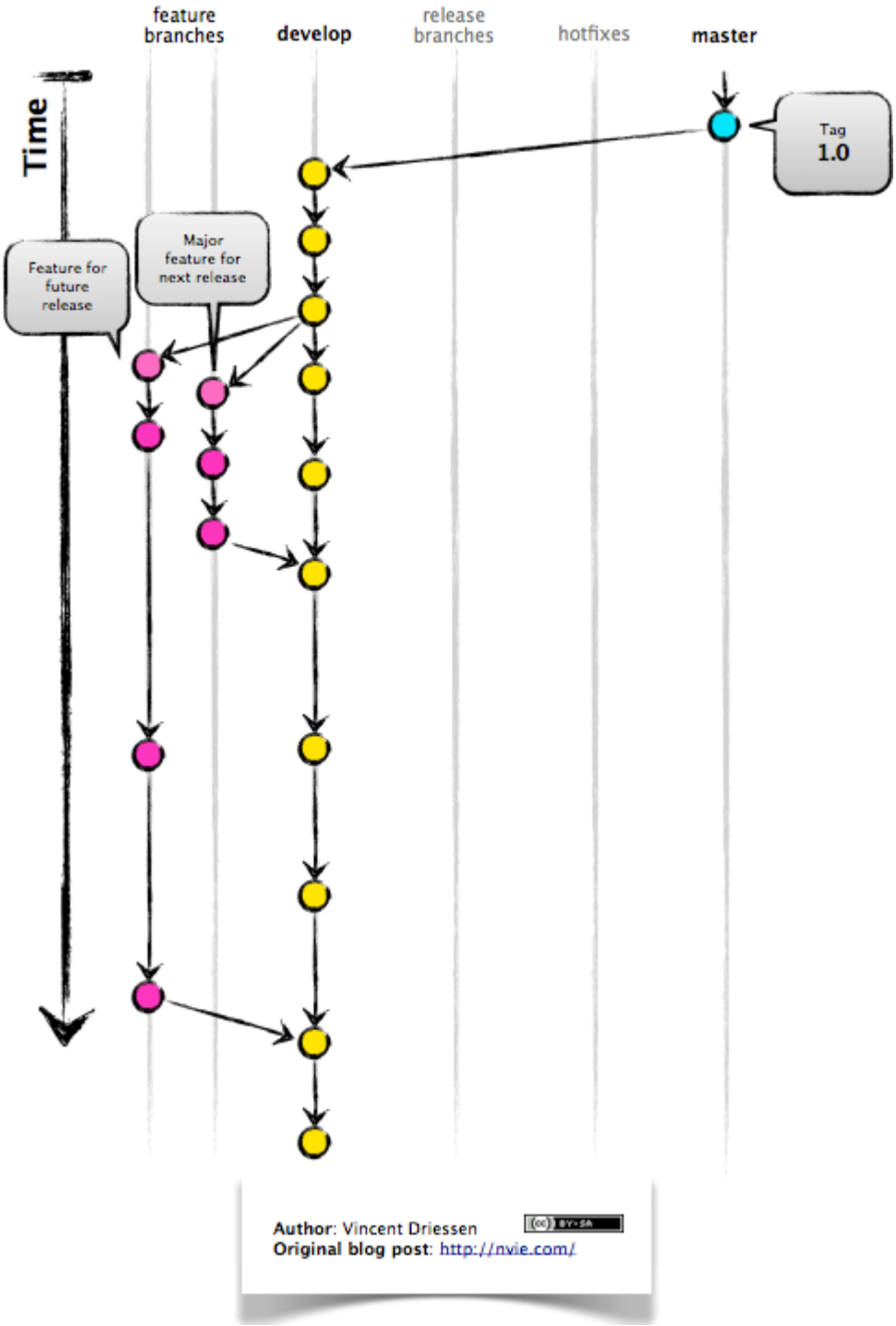
When you want to keep a linear commit history

12. Explain the Gitflow workflow with neat diagram?

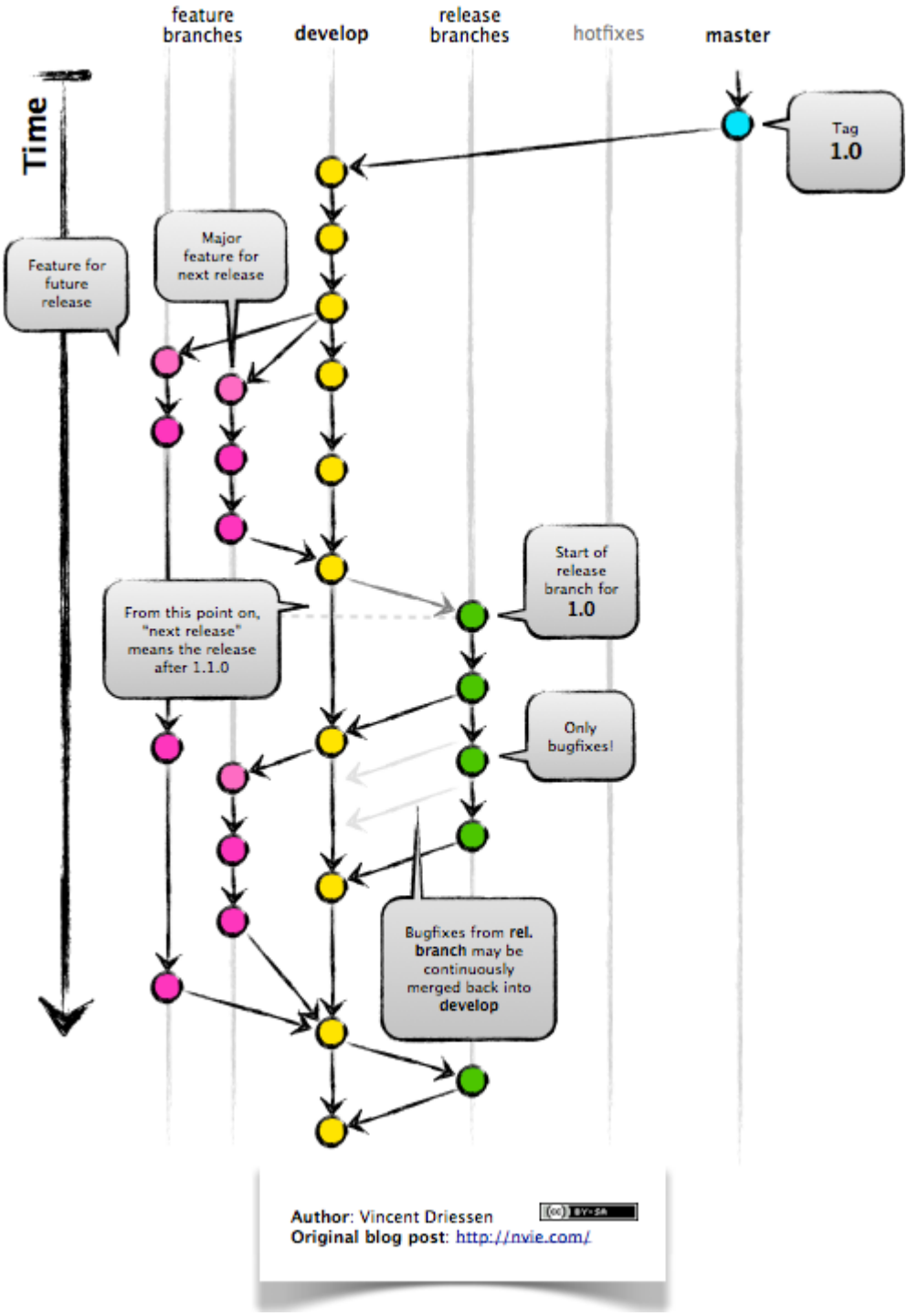
New development (new features, non-emergency bug fixes) are built in feature branches:



Feature branches are branched off of the develop branch, and finished features and fixes are merged back into the develop branch when they're ready for release:

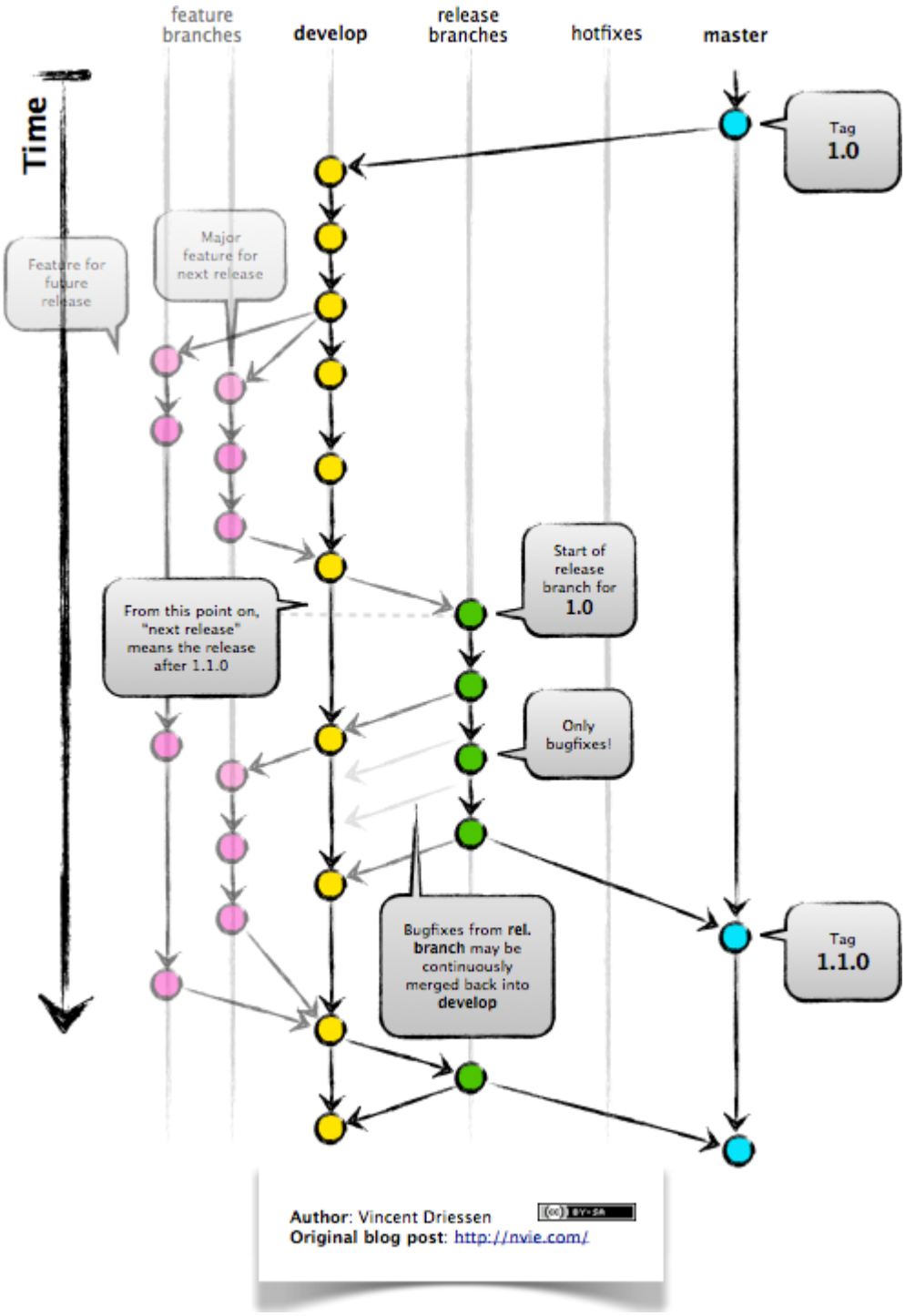


When it is time to make a release, a release branch is created off of develop:



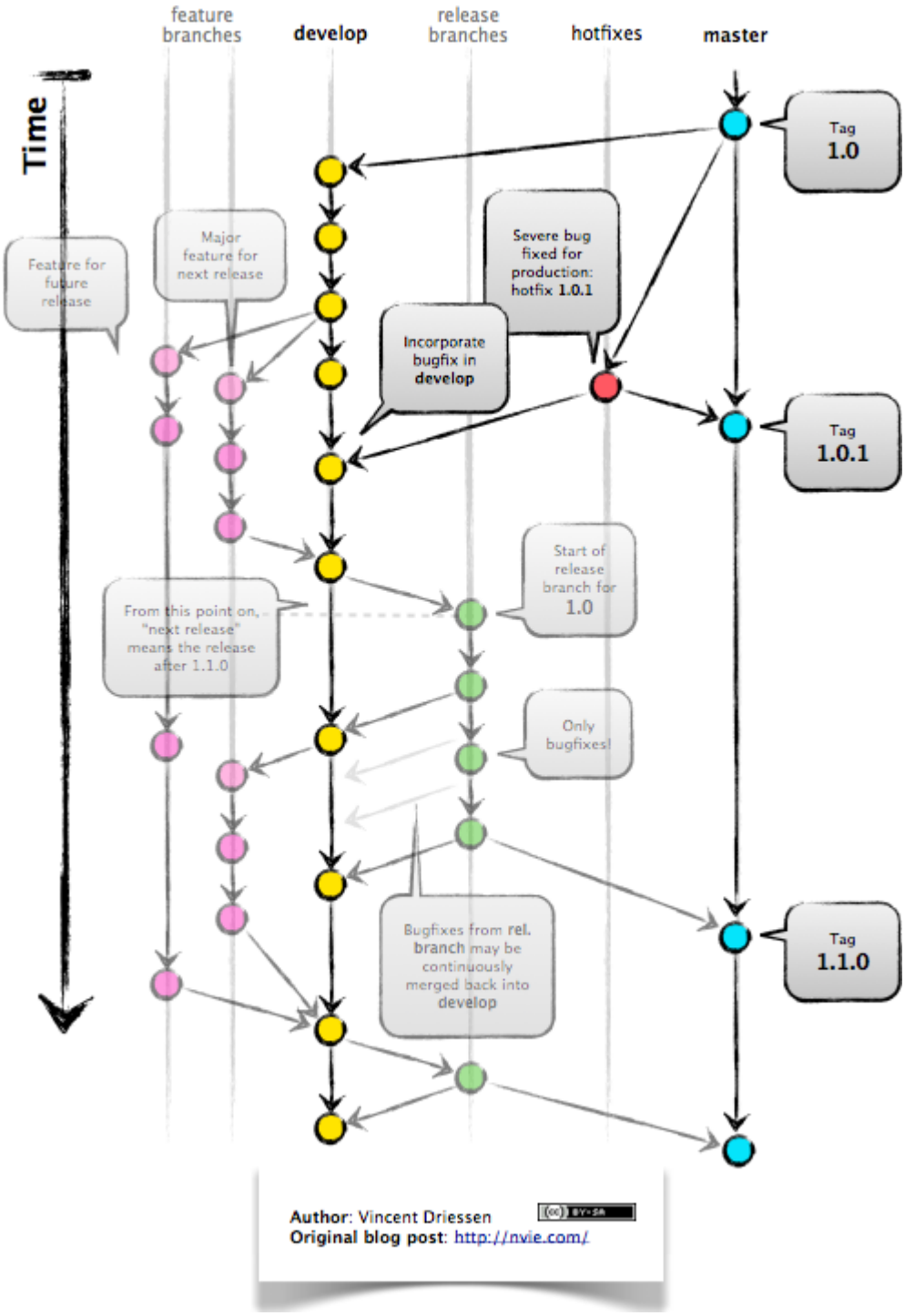
The code in the release branch is deployed onto a suitable test environment, tested, and any problems are fixed directly in the release branch. This deploy -> test -> fix -> redeploy -> retest cycle continues until you're happy that the release is good enough to release to customers.

When the release is finished, the release branch is merged
into master and into develop too, to make sure that any changes made in the release branch aren't accidentally lost by new development.



The master branch tracks released code only. The only commits to master are merges from release branches and hotfix branches.

Hotfix branches are used to create emergency fixes:



They are branched directly from a tagged release in the master branch, and when finished are merged back into both master and develop to make sure that the hotfix isn't accidentally lost when the next regular release occurs.

13. How can we see 'n' most recent commits in GIT?

git log -n 10 –author=<your name>

14. How can we know if a branch is already merged into master in GIT?

git branch –merged master

15. What is the most popular branching strategy in GIT?

Gitflow

16. How to delete a branch in Git?

git branch -d <name of branch>

git branch —delete <name of branch>

17. How do you clean unwanted files from working directory in Git?

To remove directories, run git clean -f -d or git clean -fd.

To remove ignored files, run git clean -f -X or git clean -fX.

To remove ignored and non-ignored files, run git clean -f -x or git clean -fx.

18. How can we determine the commit that is the source of bug in Git?

The slowest, most tedious way of finding a bad git commit is something we've all done before. You checkout some old commit, make sure the broken code isn't there, then checkout a slightly newer commit, check again, and repeat over and over until you find the flawed commit.

19. How can we see differences between two commits in Git?

git diff <commit-SHA-id> <commit-SHA-id>

20. What is cherry-pick in Git?

Cherry picking is the act of picking a commit from a branch and applying it to another.

21. What is the use of Git log command?

The Git Log tool allows you to view information about previous commits that have occurred in a project.

22. What is the purpose of stash command in Git?

Stashing takes the dirty state of your working directory — that is, your modified tracked files and staged changes — and saves it on a stack of unfinished changes that you can reapply at any time

23. What is the use of status command in Git?

The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git. Status output does not show you any information regarding the committed project history.

24. Explain the use of git rerere?

The git rerere functionality is a bit of a hidden feature. The name stands for “reuse recorded resolution” and, as the name implies, it allows you to ask Git to remember how you've resolved a hunk conflict so that the next time it sees the same conflict, Git can resolve it for you automatically.

25. Name atleast 5 GUI used for working on Git?

a. GitHub
b. GitLab
c. QGit
d. Gitg
e. Git Force
f. Sourcetree

g. GitUp
h. GitBox