Git

Gitflow // GIT // Github flow

Korte opfrissing

- Version control
- Collaboration
- Bijhouden van changes versus files
- Gemakkelijk stappen terug
- Elk eigen versie

- Main vs Master

Branches

- Aftakkingen van het main project
- Elke developer eigen branch(?)

- Develop/feature/release/bug/hotfix/…

- Branch → merge

  of

- Branch → Rebase

Lifecylce

- Normale lifecycle van een git commit

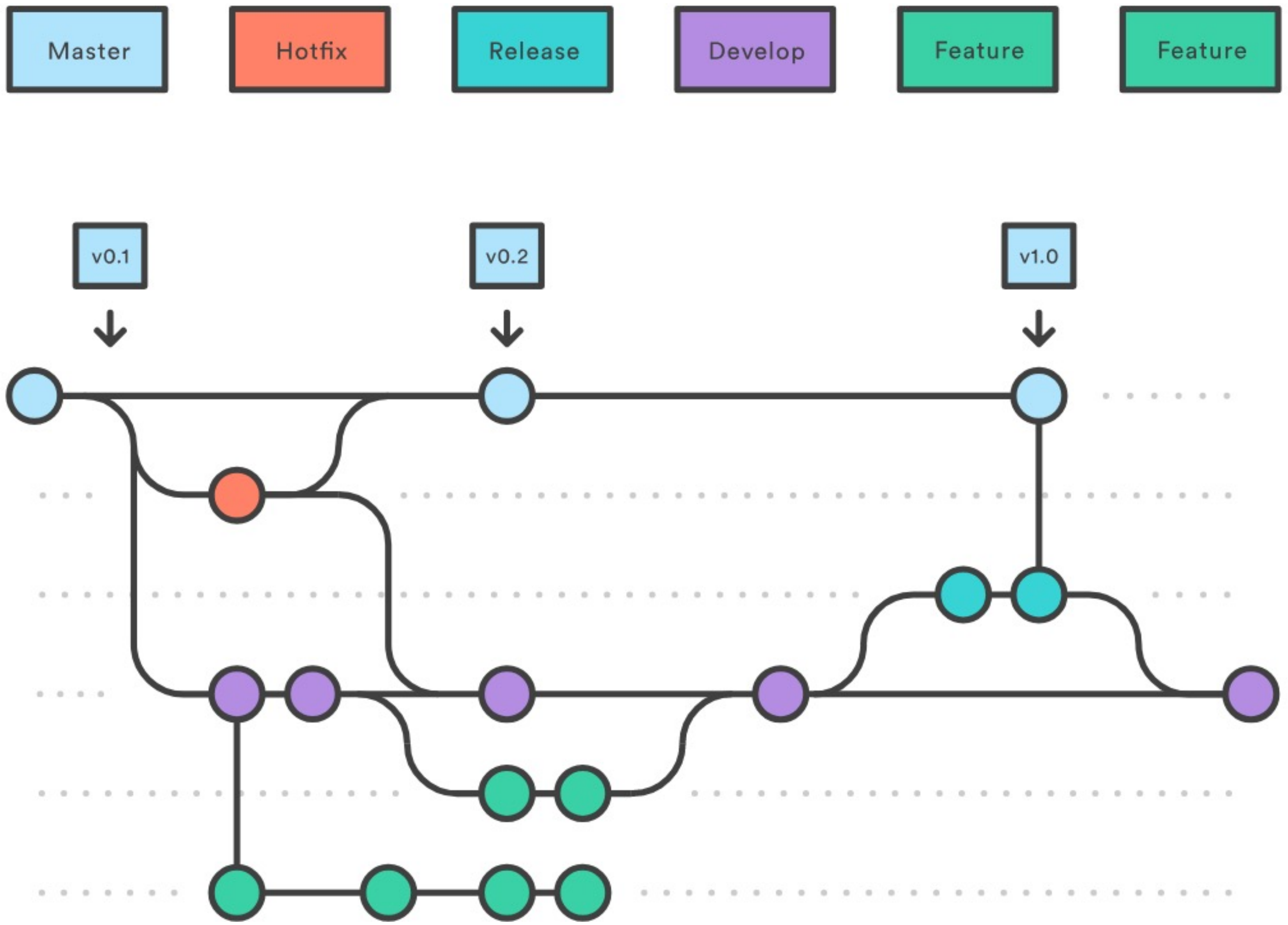
- *`git init`*
- `git add .` → `staging`
- `git commit -m "<type>(<module>): <message>"`
- `git pull`
- `git push`
- `Git merge develop`
- PR

PR requests

- Feedback vragen aan andere programmeurs
- Reorganise code en commits
- Merge into Main

- Altijd eerst main zelf mergen in eigen branch

Gitflow



https://medium.com/@rafavinnce/gitflow-branch-guide-8a523360c053

Git flow concept

- *Main branches:*
    - *master*
    - *develop*
    - *features*
    - *hotfix*
    - *release*

- Only main is releaseable

Github flow

- Anything in the master branch is deployable
- To work on something new, create a descriptively named branch off of master (ie:new-oauth2-scopes)
- Commit to that branch locally and regularly push your work to the same named branch on the server
- When you need feedback or help, or you think the branch is ready for merging, open a pull request
- After someone else has reviewed and signed off on the feature, you can merge it into master
- Once it is merged and pushed to ‘master’, you can and *should* deploy immediately

Git flow VS github flow

- Afspraken rond branches en commits
- Afhankelijk van bedrijf tot bedrijf
- Great discussion

Cycle

- Product Owner (PO) -> afspreken wat gemaakt moet worden → acceptance test
- PO maakt tickets -> as a user, I can…,
- **Dev krijgt een ticket**
- **Opent een feature branch**
- Dev: tests uitschrijven
- **Fixed**
- Refactored
- **Document**
- **Merge dev into feat**
- **PR**
- Dev -> release
- Release > main, gedeployed

Let's rewrite history

Rebase time

Useful commands

- `git log --all --oneline`
- `git rebase [branch]`
- `git rebase -i [commit]`
- `git push -f`

- `a to edit`
- `:wq to quit and write`

Rebase types

- `# p, pick <commit> = use commit`
- `# r, reword <commit> = use commit, but edit the commit message`
- `# e, edit <commit> = use commit, but stop for amending`
- `# s, squash <commit> = use commit, but meld into previous commit`
- `# f, fixup <commit> = like "squash", but discard this commit's log message`
- `# x, exec <command> = run command (the rest of the line) using shell`
- `# b, break = stop here (continue rebase later with 'git rebase --continue')`
- `# d, drop <commit> = remove commit`
- `# l, label <label> = label current HEAD with a name`
- `# t, reset <label> = reset HEAD to a label`
- `# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]`
- `# .       create a merge commit using the original merge commit's`
- `# .       message (or the oneline, if no original merge commit was`
- `# .       specified). Use -c <commit> to reword the commit message.`