How to GitFlow

nopx

April 11, 2019

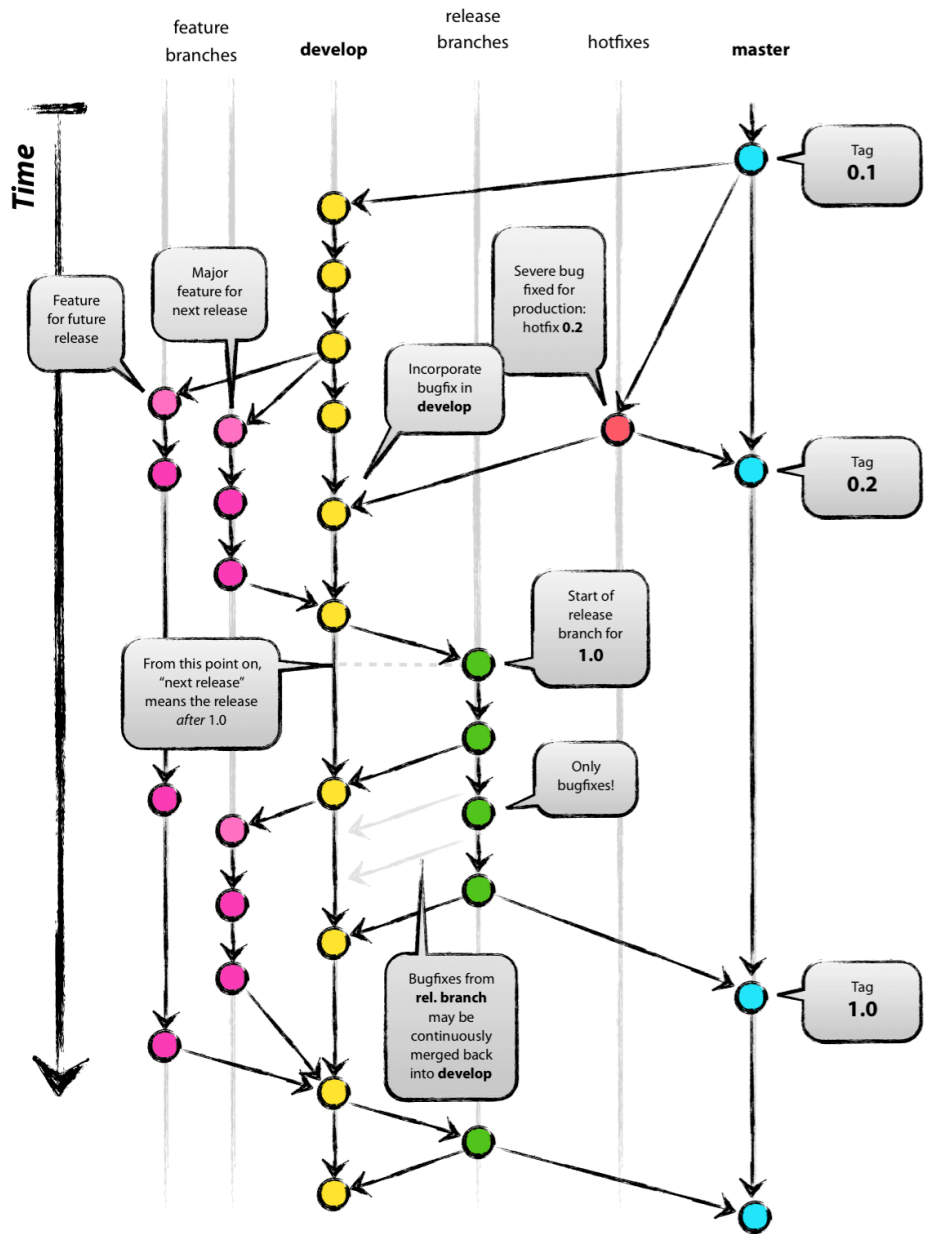
Was ist GitFlow?

- Branching-Modell
- Guideline für die Entwickler

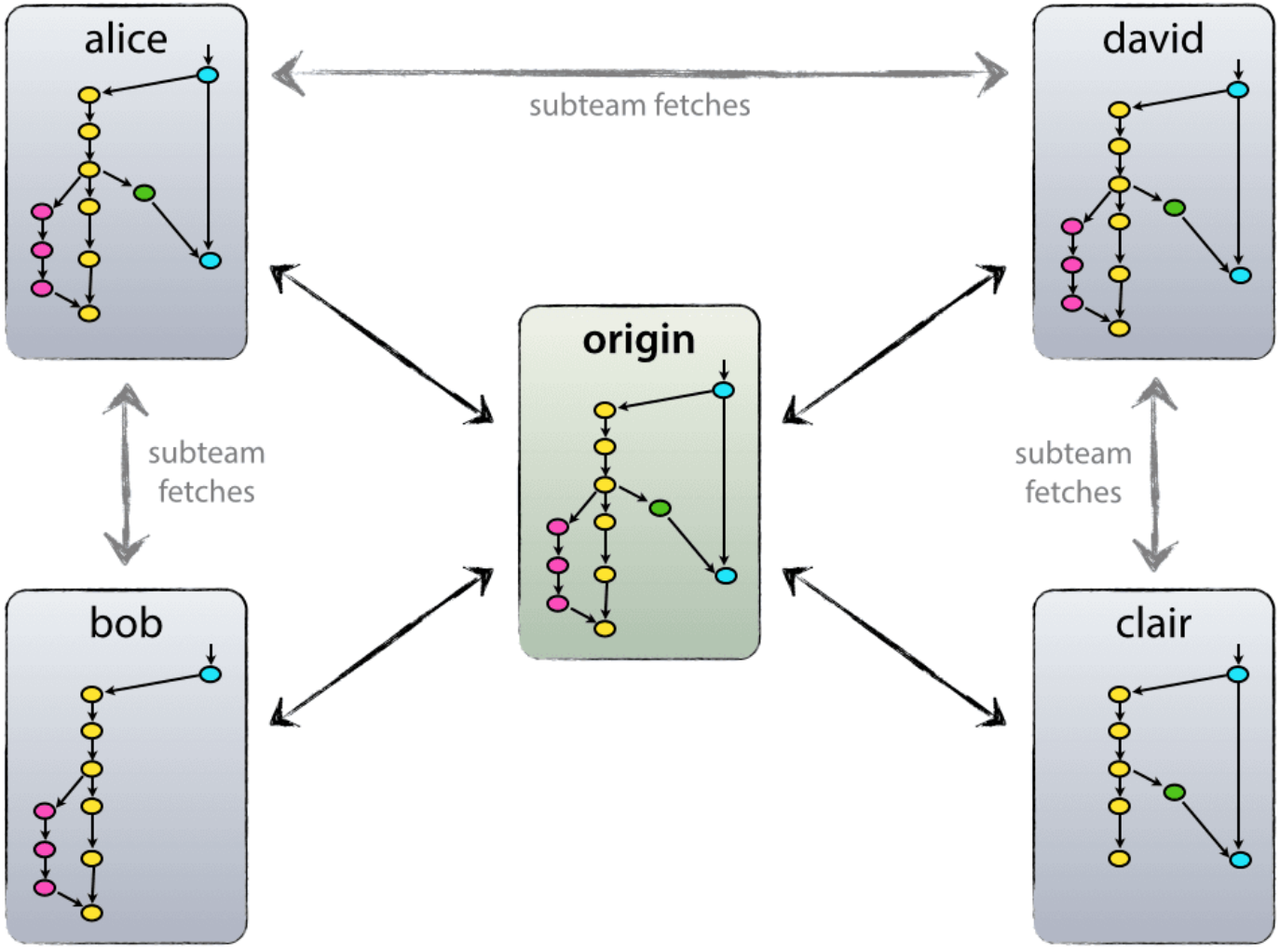
Warum brauchen wir sowas?

- *lauffähiger* Code
- Mehrere Entwickler parallel
- Reproduzierbare Stände
- Gleiches Verständnis von Branches
- Feature-Freezes

GitFlow



Dezentral versus Zentral



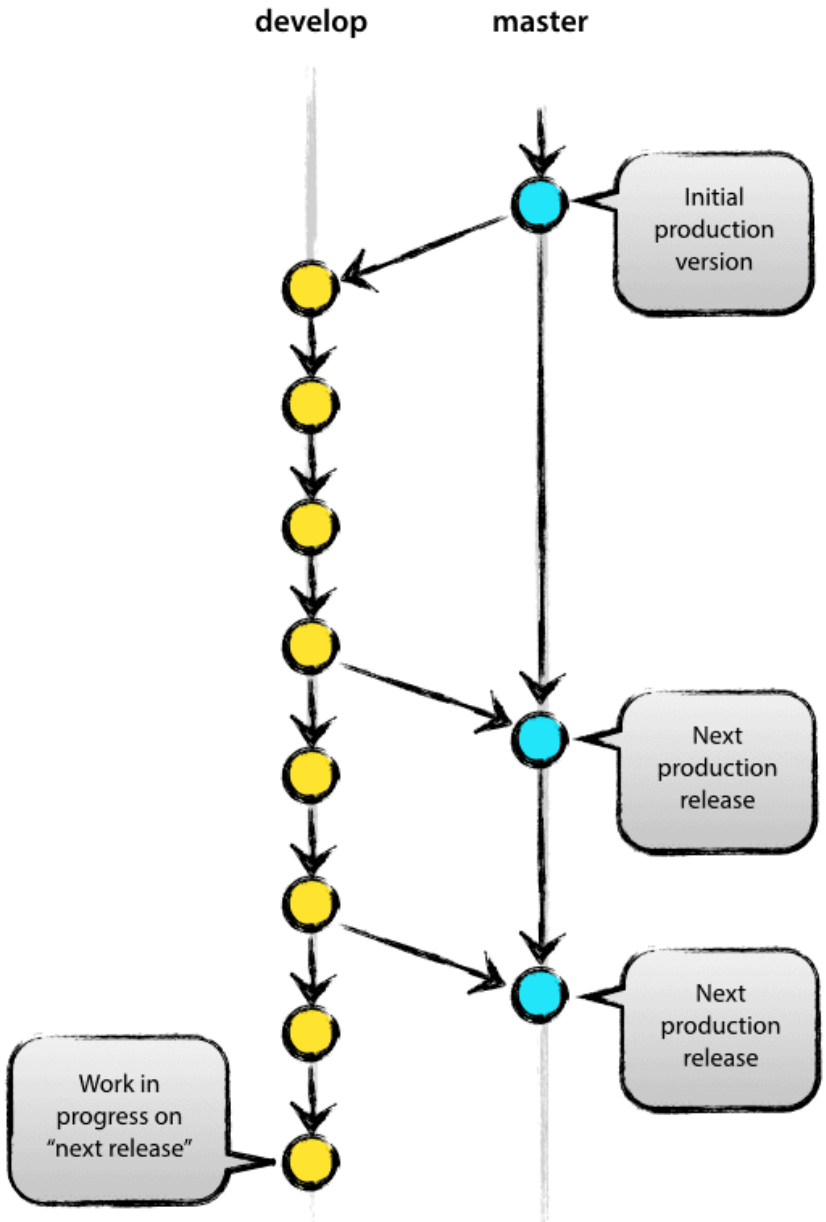
Mainbranches

- Master-Branch
  - *production-ready*
  - Namingconventions: `master`

Mainbranches

- Master-Branch
  - *production-ready*
  - Namingconventions: `master`
- Develop-Branch (Integration-Branch)
  - *Nightlybuilds*
  - Namingconventions: `develop` oder `dev`

Mainbranches



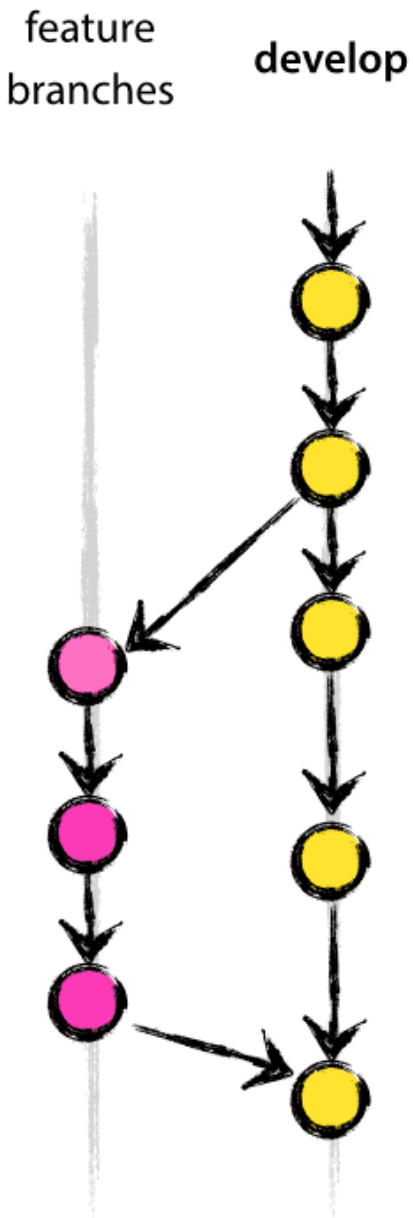
Supportbranches

- Feature-Branch (Topic-Branches)
- Release-Branch
- Hotfix-Branch

Feature-Branches (Topic-Branches)

- Branched von `develop`
- Merged in `develop`
- Namingconventions: `feature-*`
- Oft nur in Entwickler-Repos, nicht in `origin`

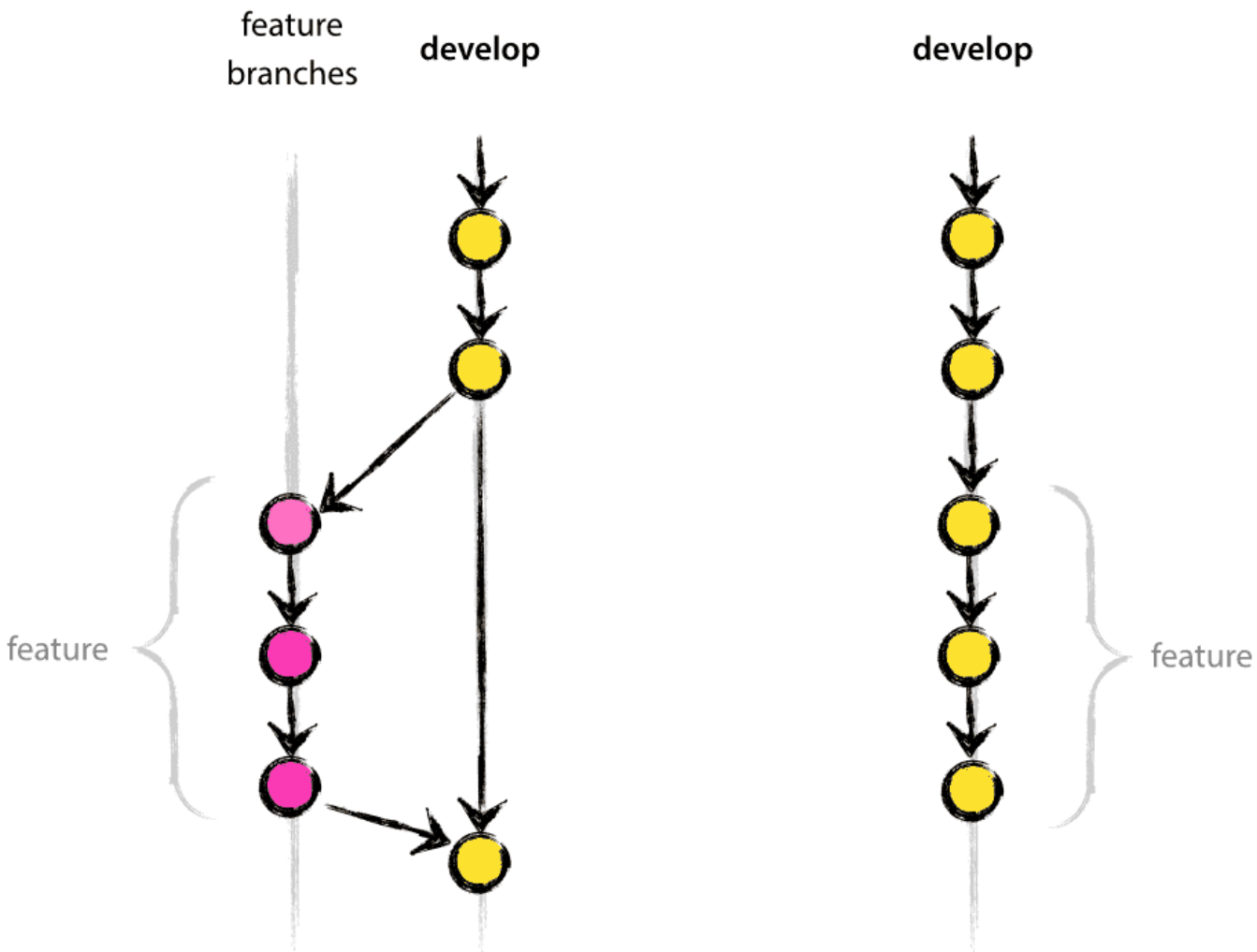
Feature-Branches (Topic-Branches)



Feature-Branches (Topic-Branches)

- Erzeuge und wechsel zu neuem Feature-Branch
  - `git checkout -b myfeature develop`
- Merge den Feature-Branch zurück
  - `git checkout develop`
  - `git merge --no-ff myfeature`
  - `git branch -d myfeature`
  - `git push origin develop`

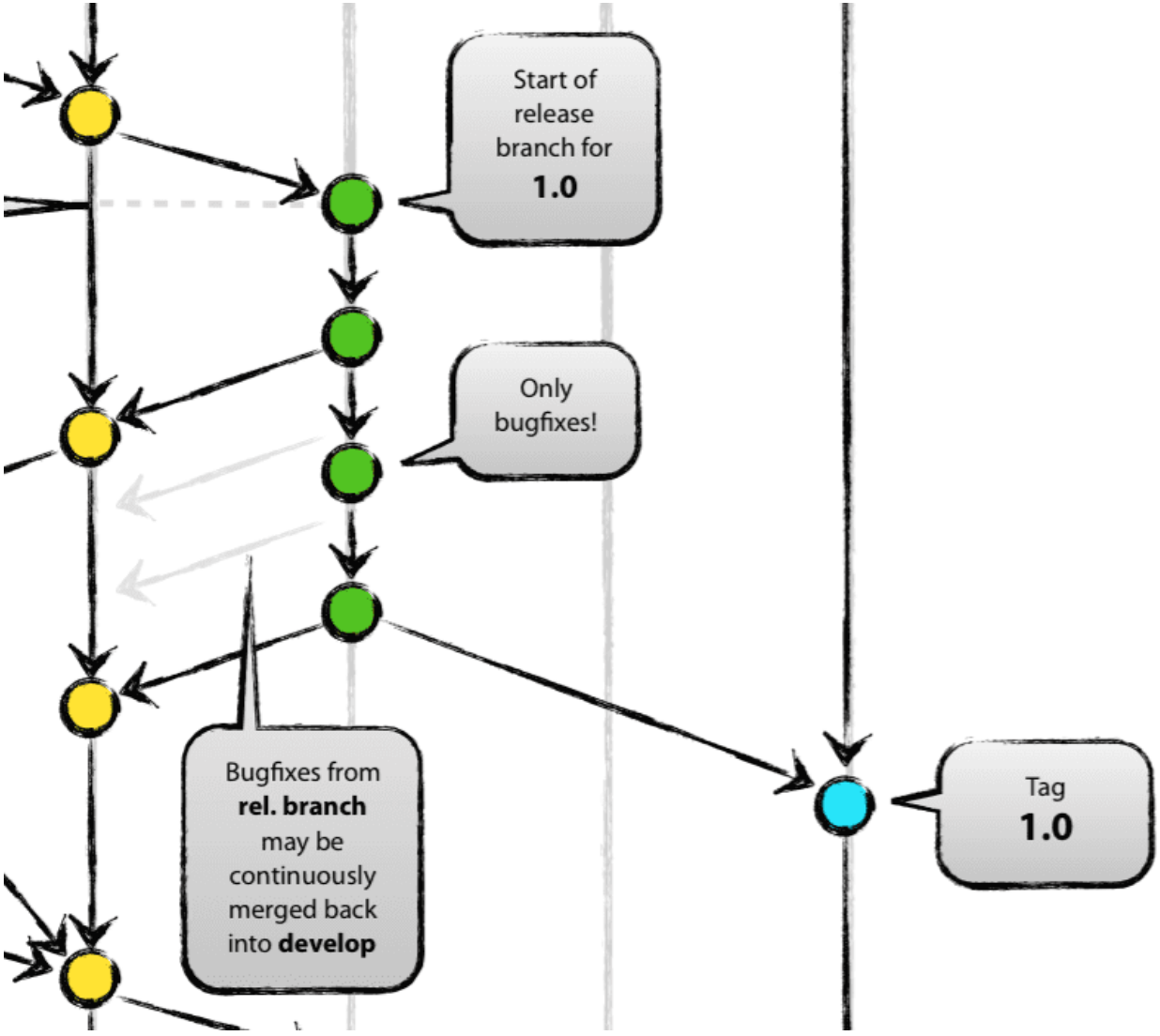
Feature-Branches (Topic-Branches)



Release-Branches

- Branched von `develop`
- Merged in `develop` oder `master`
- Namingconventions: `release-*`
- Bugfixe aber KEINE neuen Features

Release-Branches



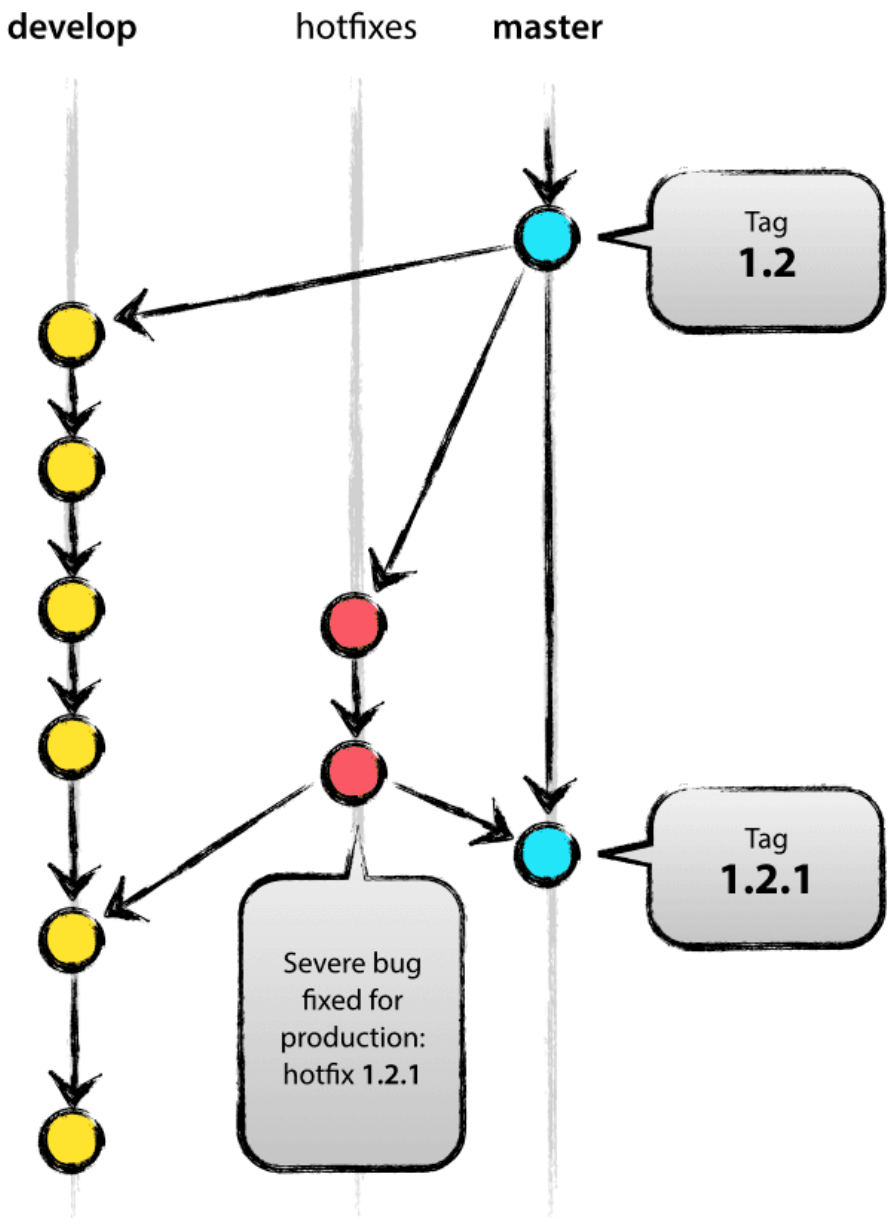
Release-Branches

- Erzeuge Release-Branch
  - `git checkout -b release-1.2 develop`
  - `./bump-version.sh 1.2`
  - `git commit -a -m Bumped version number to 1.2`
- Schließe Release-Branch ab
  - `git checkout master`
  - `git merge --no-ff release-1.2`
  - `git tag -a 1.2`
- Merge in `develop`
  - `git checkout develop`
  - `git merge --no-ff release-1.2`

Hotfix-Branches

- Branched von `master`
- Merged in `develop` oder `master`
- Namingconventions: `hotfix-*`

Hotfix-Branches



Persönliche Adaption

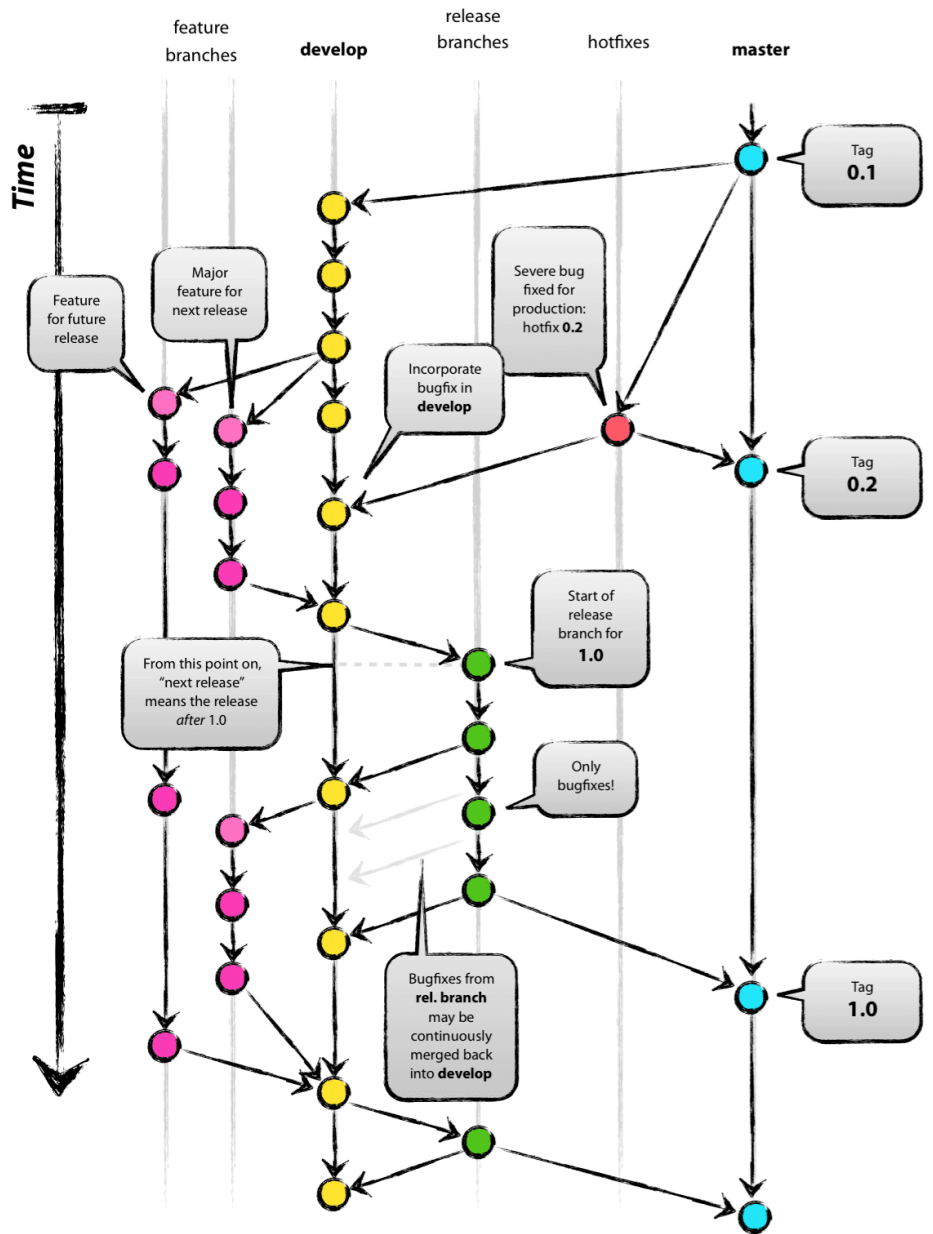
- Alpha, Beta Releases als Tags im Release-Branch
- *pump-version*-Commits auf `develop`

Zusammenfassung

- NICHT auf dem Master entwickeln!
- Größere Features in Feature-Branches
- Ein Versionierungssystem nutzen (z.B. semantic versioning 2.0.0[1])
- Versionsdatei, *pump version*-Commits
- Feature-Freezes einhalten
- `support`-Branches

[1] `https://semver.org/`

GitFlow CheatSheet



Fazit

- GitFlow muss nicht die Lösung sein!

Fazit

- GitFlow muss nicht die Lösung sein!
- Einigt euch auf eine Lösung und dokumentiert diese
- Sprecht über Dinge die nicht abgebildet werden können
- Versionsmanagement ist nicht trivial
- Jede Lösung hilft Prozesse einzuhalten
- Nutzt CI und Reviews

Quellen

- Vincent Driessen(2010) https://nvie.com/posts/a-successful-git-branching-model/
- GitFlow Cheatsheet https://danielkummer.github.io/git-flow-cheatsheet/
- https://gitversion.readthedocs.io/en/latest/git-branching-strategies/gitflow-examples/