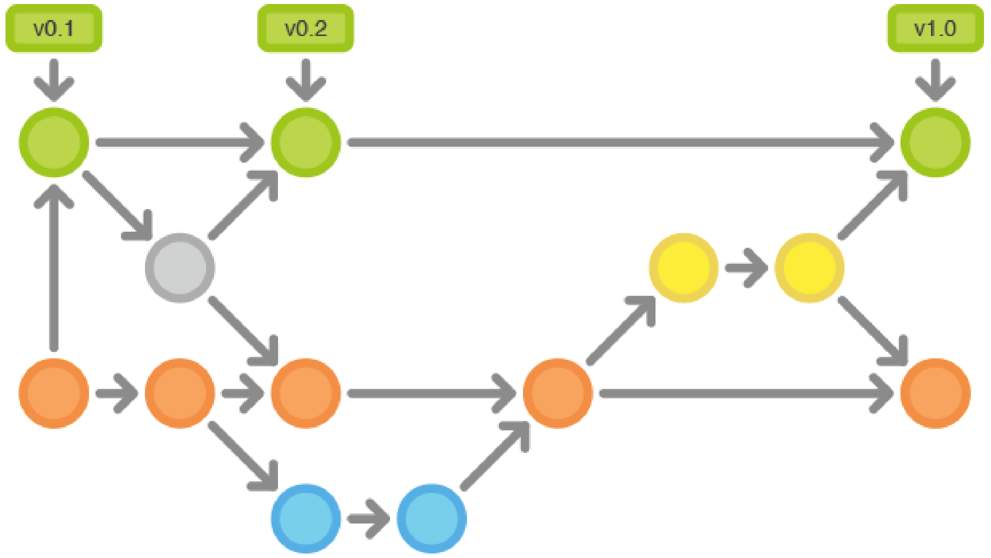
Gitflow Workflow

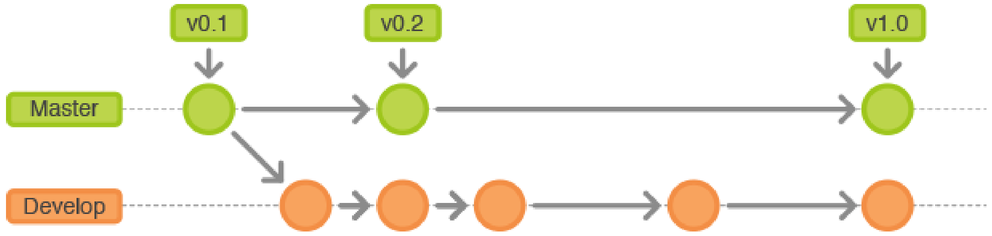
Max Kenobi

Gitflow workflow



https://leanpub.com/git-flow/read

Main branches



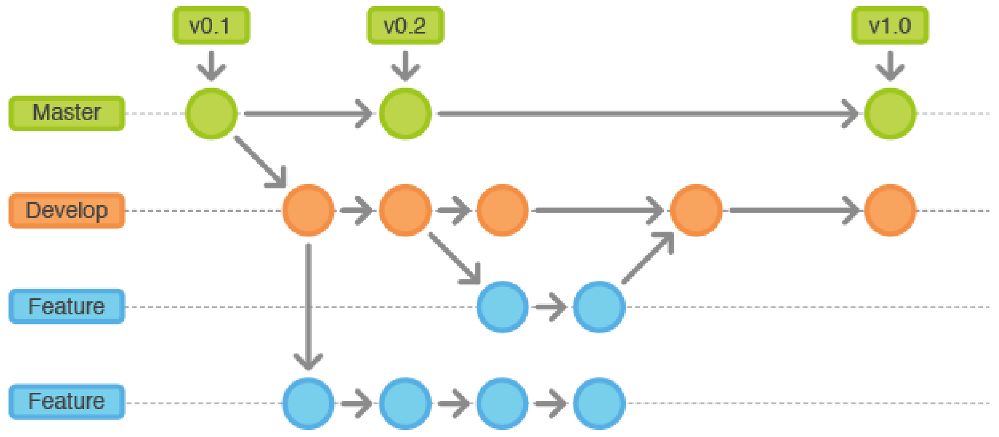
master branch

- contains *production-ready code* that can be released
- stores the official release history
- tag all commits with a version number

develop branch

- contains *pre-production code* with newly developed features
- integration branch for new features

Feature branches

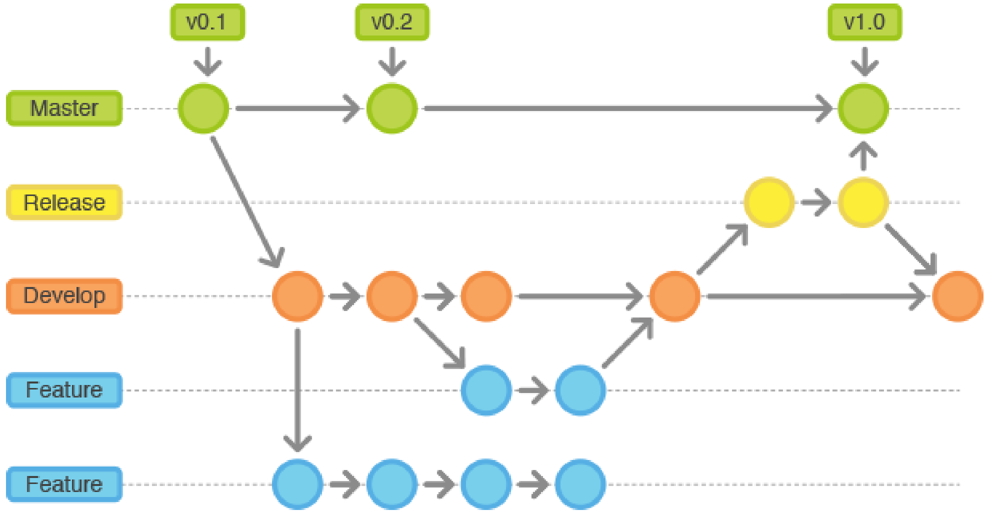


feature branch

- used when adding *new features* to your code
- the most common type of branch

https://leanpub.com/git-flow/read

Release branches

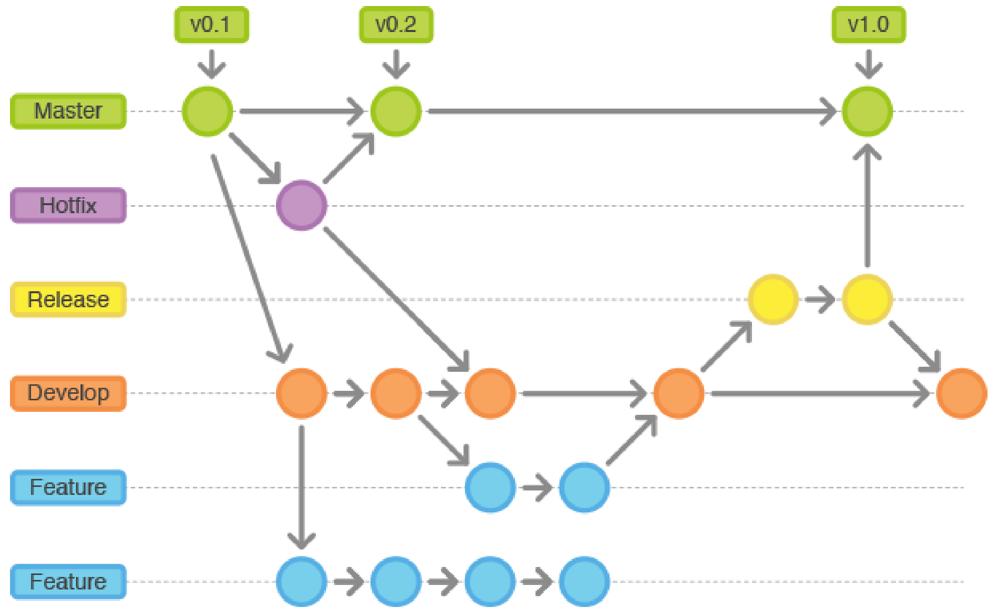


release branch

- *preparing new production releases*
- no new features, just finishing touches and minor bugs
- polish the release and work on the new features at the same time

https://leanpub.com/git-flow/read

Hotfix branches



hotfix branch

- *quickly patch production releases* (ad hoc release branches)
- address issues without waiting for the next release cycle

https://leanpub.com/git-flow/read

Gitflow extension

No need to be a git master!

Initialise gitflow and set default branch names:

```
> git flow init
```

Start/finish feature branches:

```
> git flow feature start <name>
> git flow feature finish <name>
```

Push/pull a feature branch:

```
> git flow feature publish <name>
> git flow feature pull <name>
```

Start/finish release branches:

```
> git flow release start <release>
> git flow release finish <release>
```

Start/finish hotfix branches:

```
> git flow hotfix start <release>
> git flow hotfix finish <release>
```

gitflow extension repo
how to install (easy)

Extra rules

1. **Feature branches naming convention**
   feature/<youtrack_task_id>/<short_meaningful_description>
   ex. *feature/ML-9999/add_new_awesome_feature*
2. **Tags naming convention**
   <huge_changes_num>.<new_feats_num>.<hotfix_num>
   ex. *1.10.2*, *2.0.0*, etc.
   - *huge_changes_num:* changes that affect the whole repo code
   - *new_feats_num:* the most commonly changed num, new features
   - *hotfix_num:* hotfixes; really hot; in the evenings or on weekends
3. **Code review in GitLab UI**
   - before “git flow feature finish”
   - make merge requests into the develop branch
4. **Delete old branches**

Pros and Cons

Pros

- releases
- support multiple versions
- production-ready code in master
- improved collaboration
- built-in code quality reviews
- clean git history

Cons

- complexity
- slows down the deployment

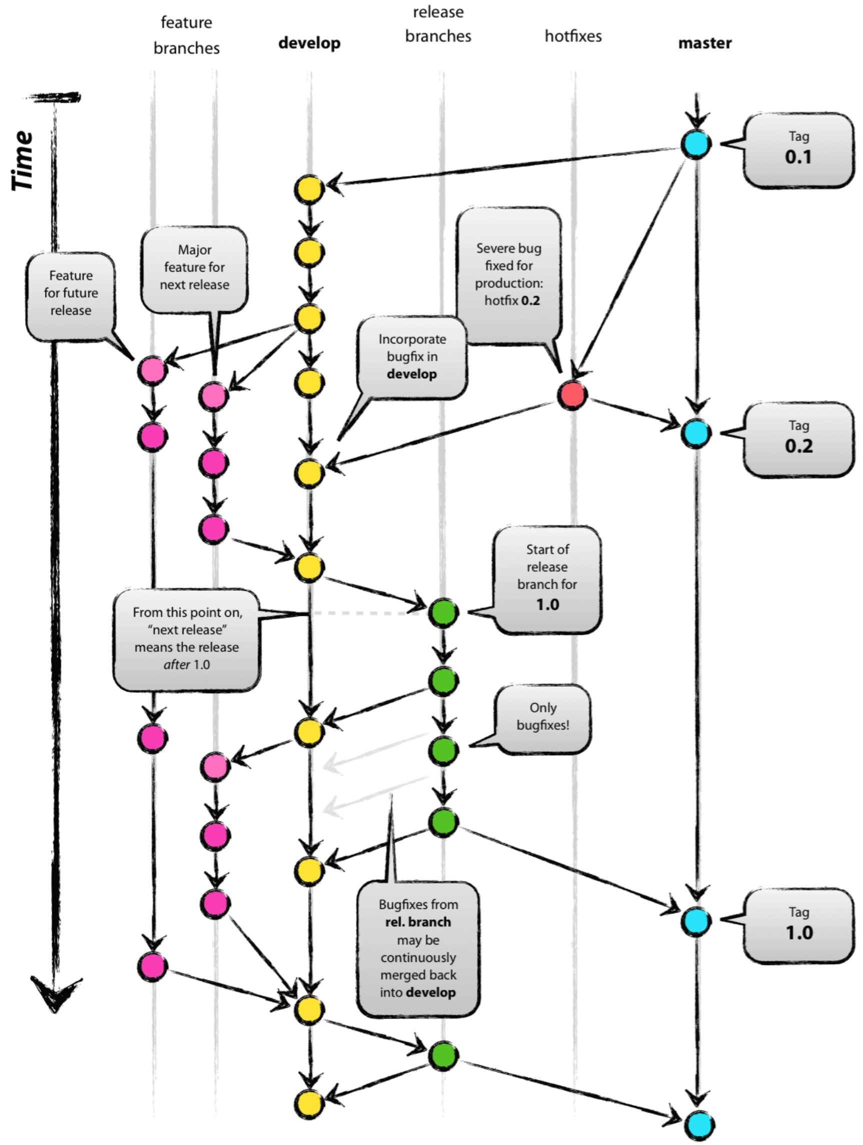
Additional links

Post by the author: https://nvie.com/posts/a-successful-git-branching-model/

https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

https://github.com/nvie/gitflow

https://leanpub.com/git-flow/read



feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

https://nvie.com/posts/a-successful-git-branching-model/