NETCOMPANY

GIT – FÅ STYR PÅ UDVIKLINGEN!

Date: 2020-05-06
Version: 1.5
Author: Rasmus Bækgaard
Contact: rab@netcompany.com

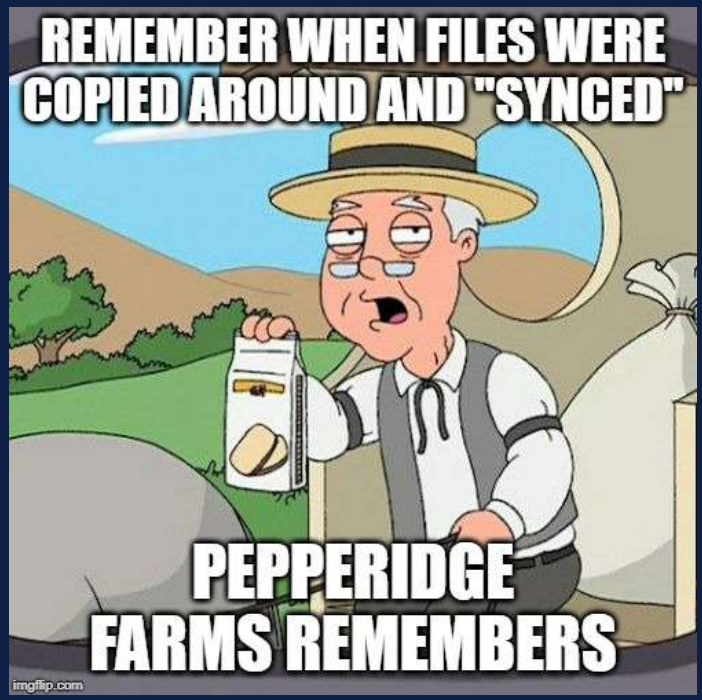
netcompany

Agenda and focus

- What is Git
  - Practical theory
  - Commands frequently used, and commit messages
- Branches
  - Merging and merge conflicts (live demo)
  - Rebasing
  - Pull Requests (live demo)
- Strategies - How most projects are (/should be) using Git
- Goodies
- Questions

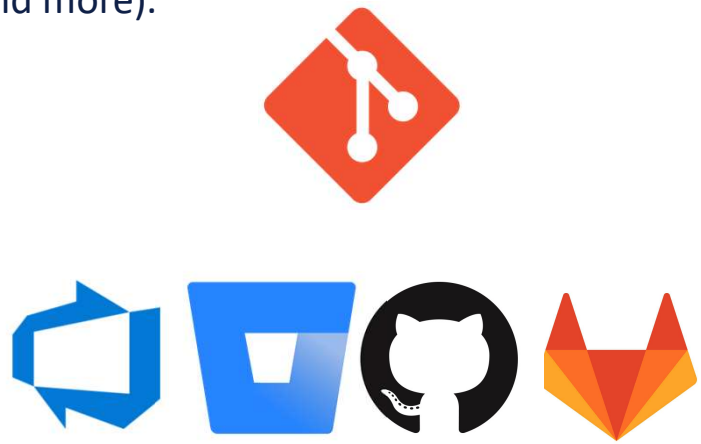
History

- USB-sticks
- .zip and .tar.gz
- Dropbox
- SVN/TFVS



Misconception: Git is not a hosting service, and not a build agent

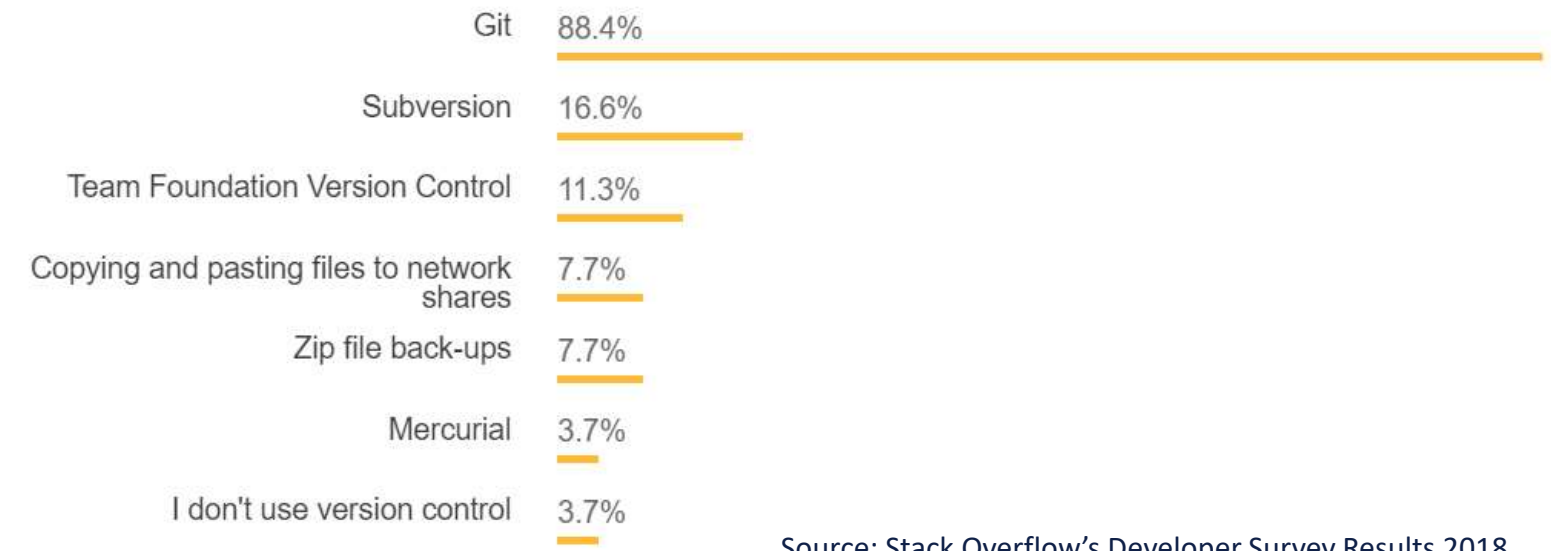
- Git is a tool to track your code.
- A hosting service is a place you keep your code (and more).
  - Azure DevOps / TFS
  - Bitbucket
  - GitHub
  - GitLab



- Most of these slides are only about the tool Git.

Who uses what?

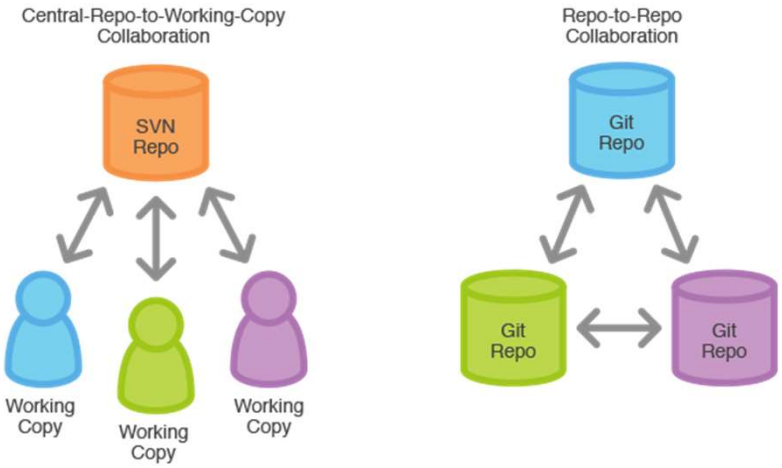
All Respondents | Professional Developers



Source: Stack Overflow's Developer Survey Results 2018

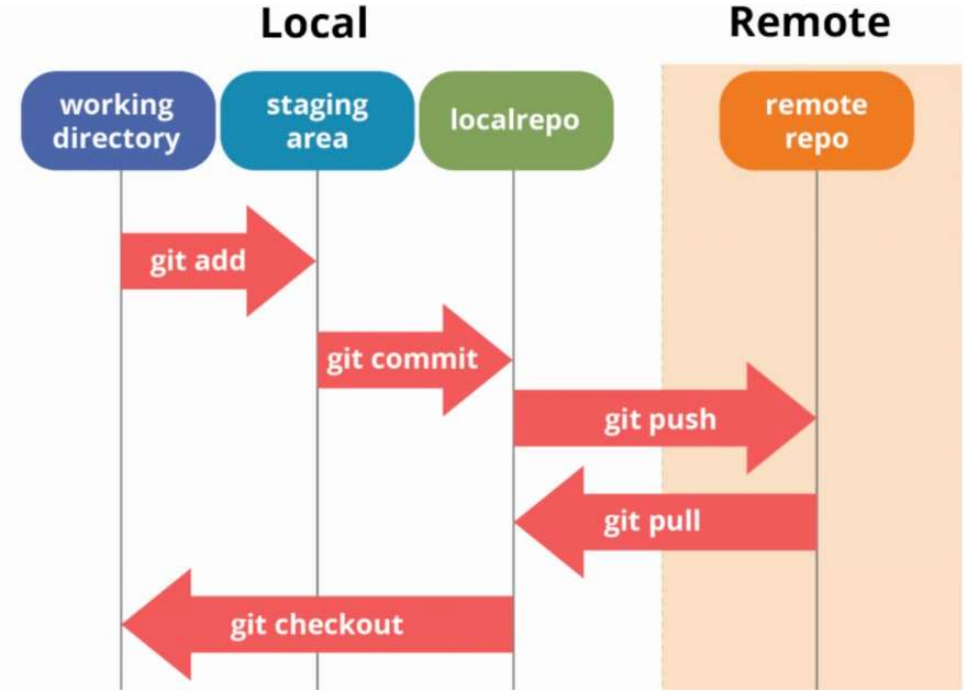
The big change with a Git repository

- You can work offline.
- The mandatory synchronization is gone.
- You can make as many “commits” as you like without interrupting your colleagues’ work.

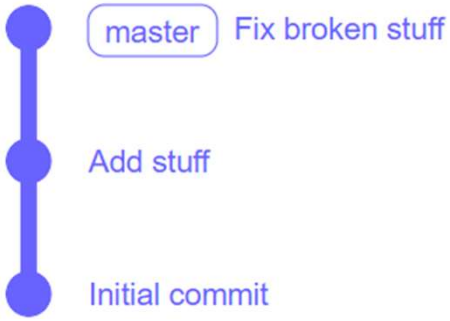


Basic commands and functionality

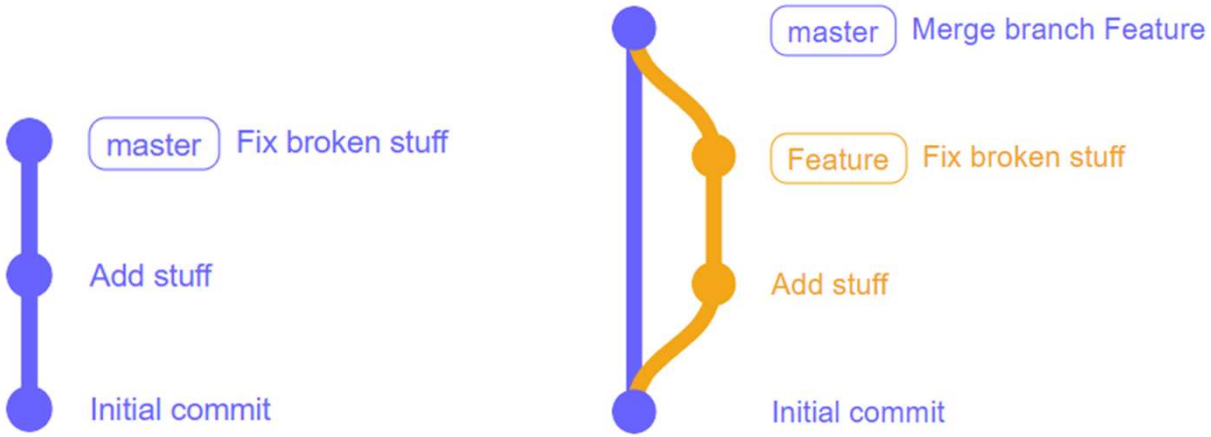
- Git introduces new commands to use:
  - Add
  - Commit
  - Fetch and Pull
  - Push
  - Branch
  - Checkout (switch branch)
  - Merge / Rebase
  - Stash
- I have a cheat sheet you can get after the talk, don't worry.



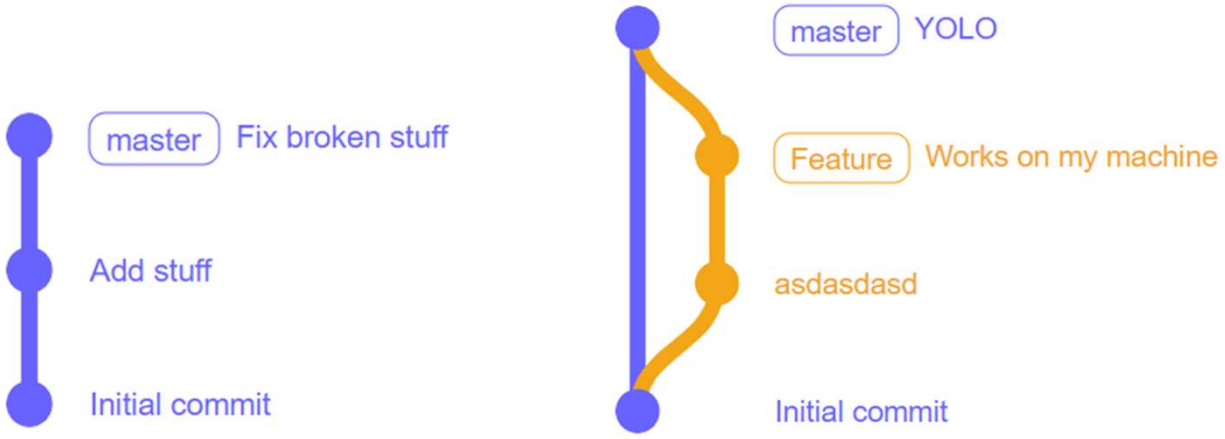
Branches and merges – the overview



Branches and merges – the overview



Branches and merges – Commit message



Commit messages

- Why write the message?
  - Speed up the reviewing process.
  - The reviewer might not understand why you made a change.

- How to write the message?
  - Ask yourself “Why did you make the commit?”
  - Imagine finishing the line “This commit will…“
    - ‘Add’ / ‘Fix’ / ‘Remove’ / …
  - The smaller the impact, the better.

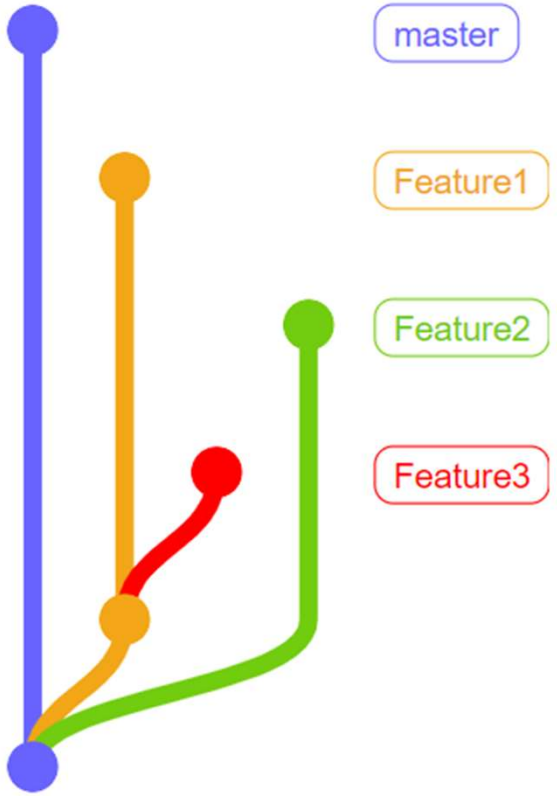
PAUSE

Du er velkommen til at stille spørgsmål i chatten

Branches

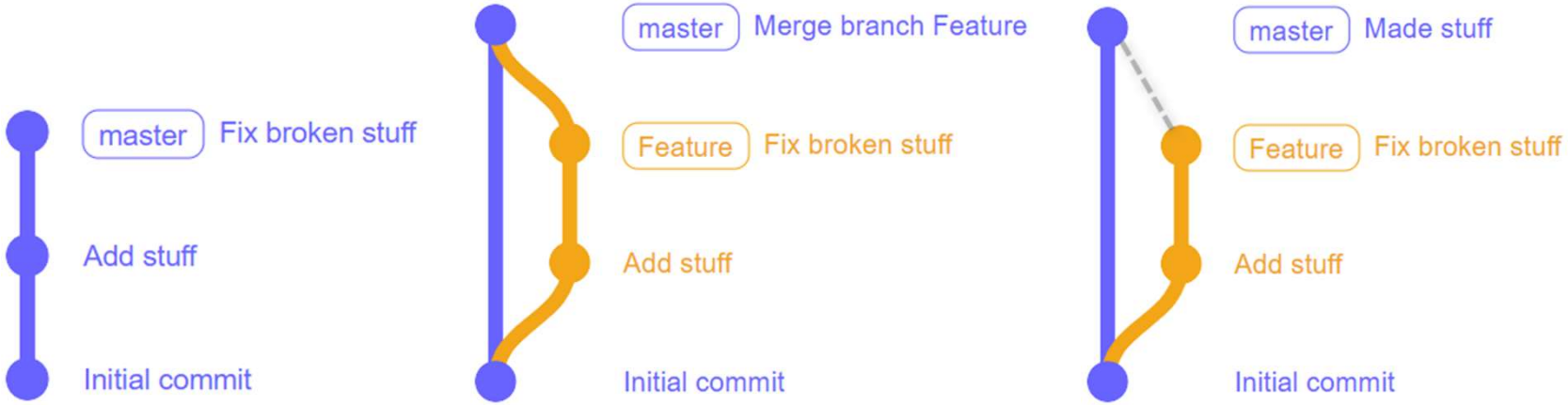
Branches

- Encapsulate what you are working on.
- Should have only one purpose.
  - Make them short-lived.
- Work on multiple features if you need to.
- Don’t worry about other developers’ code.
- Naming:
  - Feature/<Case ID>_<Title>
  - Hotfix/<Title>
  - Release/<Release ID>



Merging

- Get one branch's content into another
- Three variants of merging:
  - Fast-forward
  - No fast-forward
  - Squash merge



Branches and merge conflicts

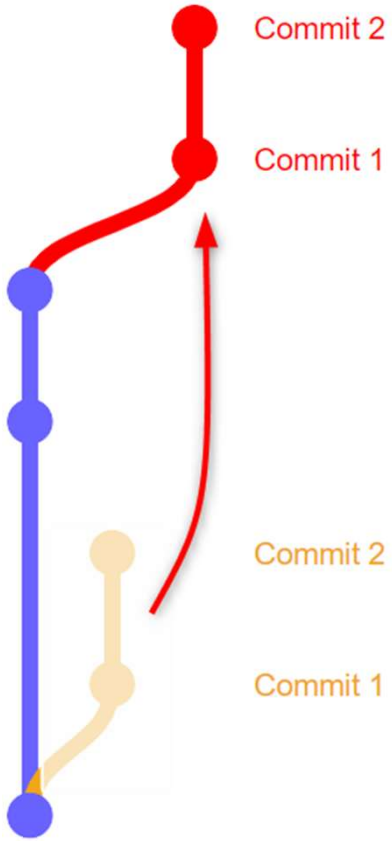
- **Live demo**

PAUSE

Du er velkommen til at stille spørgsmål i chatten

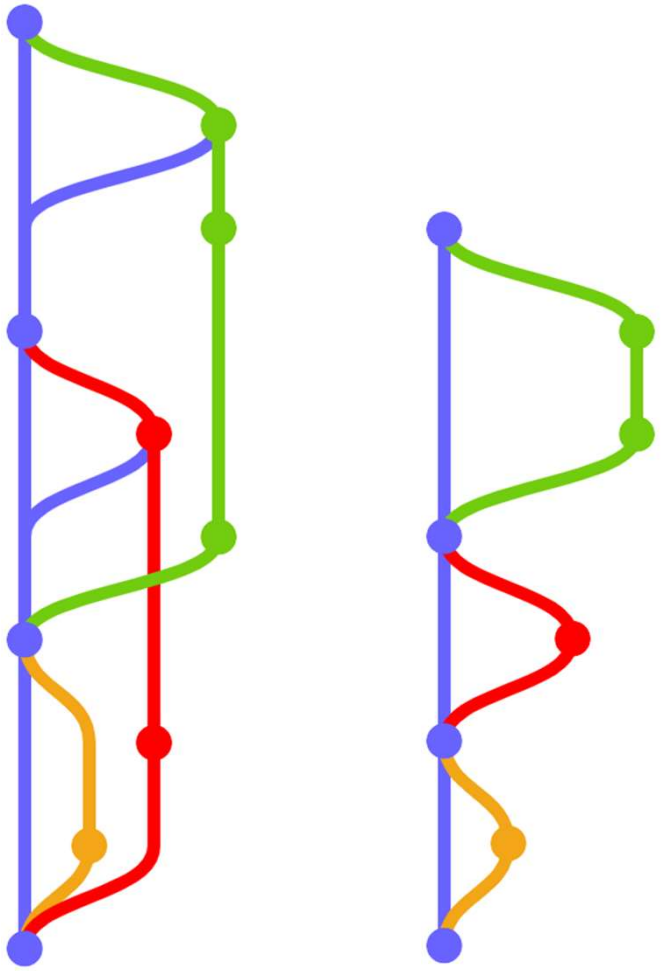
Rebase

- Alternative to merging into your feature.
- Takes a copy of the commits and moves them up.
- Takes a bit getting used to.



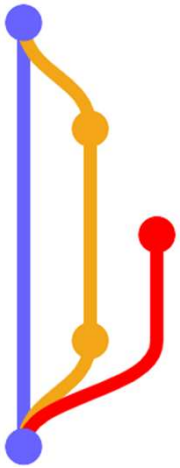
Rebase – Best idea ever

- Make non-linear history linear
- Gives a nicer history.
- Can also modify the commits:
  - Fewer commits
  - Another message

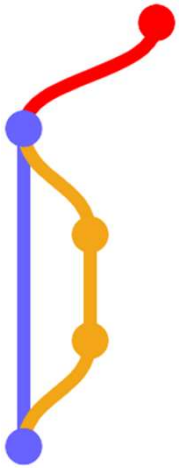


Rebase – Best idea ever

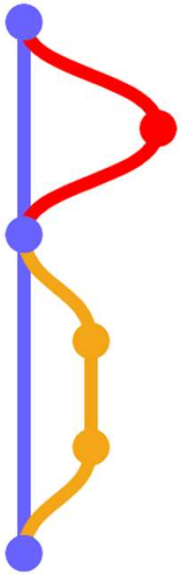
1. Red branch is based on an old master



2. Red branch is rebased on the latest master



3. Red branch is merged into master



Rebase – Best idea ever

GOLDER RULE OF REBASING:

NEVER EVER rebase a branch that has been pushed.

Rebase – Best idea ever

GOLDER RULE OF REBASING:

NEVER EVER rebase a branch that has been pushed.

- Rebasing makes new commits, with new content -> thereby new parents.
- Duplicate of the parents in both branches.

Pull Requests

- Merging a Feature into Development/Releases/Master

- This is your code review.
- This is your ”can it still build”-check.
- This is your ”do all the unit tests flag green?”-check.

- **Live demo**

PAUSE

Du er velkommen til at stille spørgsmål i chatten

Workflow – What strategy to use?

- Release & Trunk (Master & Development)
  - Latest release, and what is under development.

Workflow – What strategy to use?

- Release & Trunk (Master & Development)
  - Latest release, and what is under development.

- Multi-Release & Trunk (Master, Release 1.6, Release 1.7, Release 1.8, Development)
  - Multiple branches with releases, and a branch for development.

Workflow – What strategy to use?

- Release & Trunk (Master & Development)
  - Latest release, and what is under development.
- Multi-Release & Trunk (Master, Release 1.6, Release 1.7, Release 1.8, Development)
  - Multiple branches with releases, and a branch for development.
- Git Flow (Master, Release 1.6, Release 1.7, Release 1.8, Development, Feature/X, Feature/Y…)
  - A branch for each release.
  - A branch for each issue / case / feature.
  - Merge with Pull Request

Workflow – What strategy to use?

- Release & Trunk (Master & Development)
  - Latest release, and what is under development.
- Multi-Release & Trunk (Master, Release 1.6, Release 1.7, Release 1.8, Development)
  - Multiple branches with releases, and a branch for development.
- Git Flow (Master, Release 1.6, Release 1.7, Release 1.8, Development, Feature/X, Feature/Y…)
  - A branch for each release.
  - A branch for each issue / case / feature.
  - Merge with Pull Request
- Git Flow with Blessed Repository
  - Each team is developing their own version, and these will later be merged.

Workflow – What strategy to use?

- Release & Trunk (Master & Development)
  - Latest release, and what is under development.

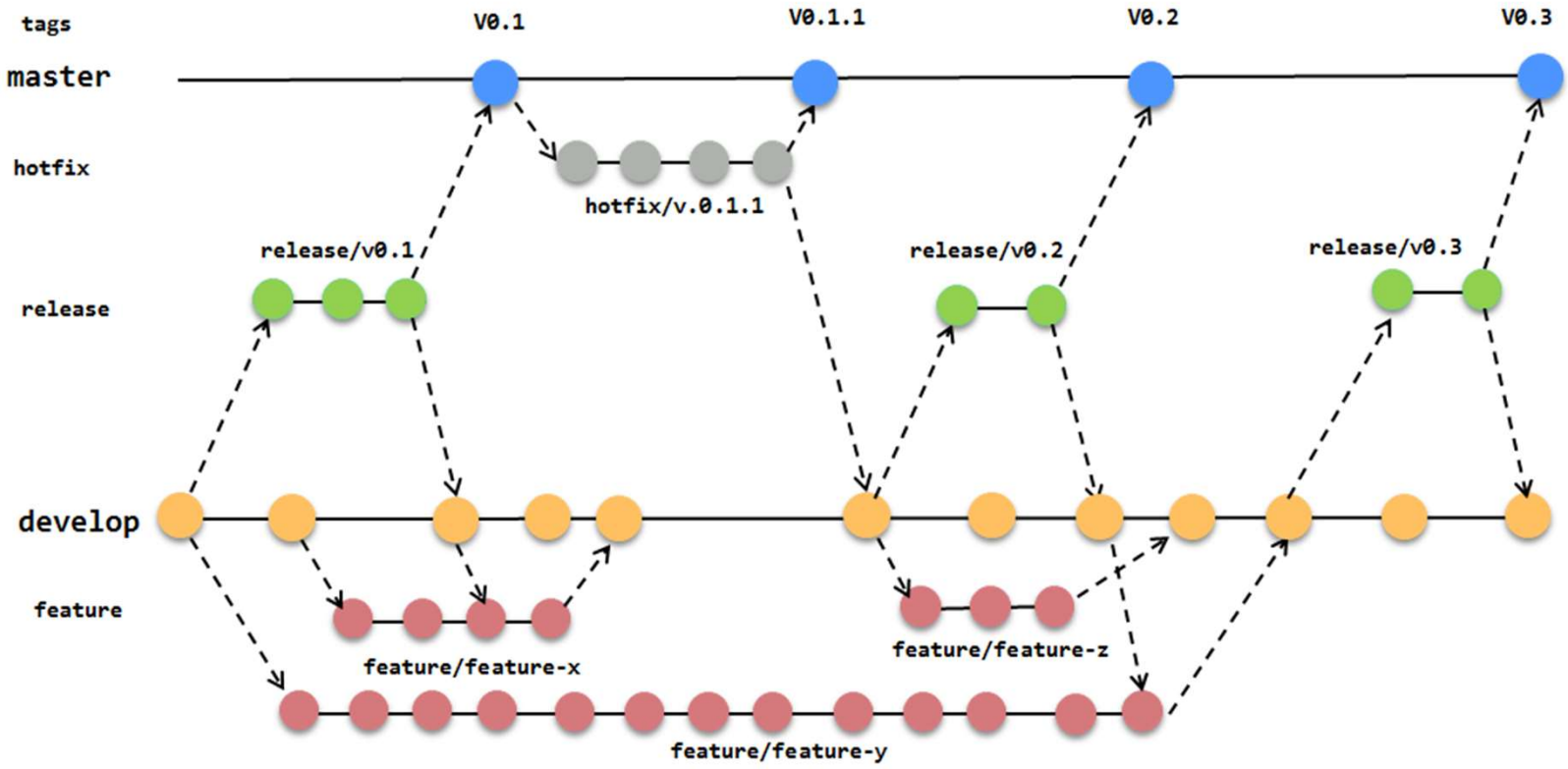
- Multi-Release & Trunk (Master, Release 1.6, Release 1.7, Release 1.8, Development)
  - Multiple branches with releases, and a branch for development.

- **Git Flow (Master, Release 1.6, Release 1.7, Release 1.8, Development, Feature/X, Feature/Y…)**
  - **A branch for each release.**
  - **A branch for each issue / case / feature.**
  - **Merge with Pull Request**

- Git Flow with Blessed Repository
  - Each team is developing their own version, and these will later be merged.

Branches – in Git Flow

Workflow – Git Flow



Git good – good practices

- When starting a new issue/ticket/bug/feature, **create a new branch**
  - You can switch to another if need be, and back again.
- **Commit often**
  - When added a new test / refactored code in file / before doing something “risky”.
  - And make a meaningful commit message *every* time.
- **Push before you go home**.
  - Avoid lost work, and your branch is you own anyway.
- Merge only from a feature branch to the development branch using **Pull Requests**
  - You review your own code and add additional information if something should happen in the release notes.
  - Someone else will read the code and add comments to it before accepting the changes.
- Delete the feature-branch when it has been merged into Master
  - Both locally and on the remote.

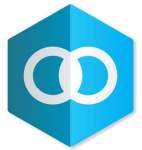
Tools

GUI Clients

- Git Extensions
- SourceTree

External Merge Tools

- P4Merge
- …



Code Editors

- Visual Studio
- IntelliJ

Command Line

- PowerShell with Git-Posh

Goodies

Setup Git, and a cheat sheet:
https://github.com/bakgaard/GitSetup

QUESTIONS?