Introduction to Git

Lars Nielsen

AARHUS UNIVERSITY

Scale IoT

NETX

Agenda

TBW

General

- Provides a common repository for code
- A system that lets you keep track of different version of file
- It allows for traceability

Most

- Enables collaboration
- Provide merging of files

Some

- Fights collaboration
- Centralised
- Distributed

- There is one repository all works on
- It is often in a server
- Some requires file lock and release
  - Some are even worse

- Every computer has a repository
- Most sync to a common base

- Released in 2005 (so "young")
- Create because according to "the git" all other VCS sucked
  - Especially CVS
- Maintain Juni Hamano

Linux

It should be in your package repository (if not switch distro).

MacOS

Should be pre-installed. But is easier to mange with Homebrew

Windows

It can be downloaded straight from: `https://git-scm.com/`

There are 6 main commands in *git* and it should be the only once you should need in the beginning.

- clone
- add
- commit
- push
- pull
- status

and *checkout*.

The *clone* command is used for getting a copy of the repository

```
git clone URI
```

- You get the full version history
- You get all current branches

```
git add [params]
git commit -m "MSG"
```

add

- adds `[params]` to a staging area
  - `.` → add everything in folder
  - `*` → add all (just like BASH)
  - `-u` → add all files all ready tracked
  - `dir` → add content of the directory dir
  - `file name` → add a file or files

commit

- Commits a your staging area to the repository history
- Takes a message
  - Make it a descriptive message

```
git pull
git push
```

pull

- Get the history in the base repository
- Get the branches in the base repository
  - Depends on the base setup

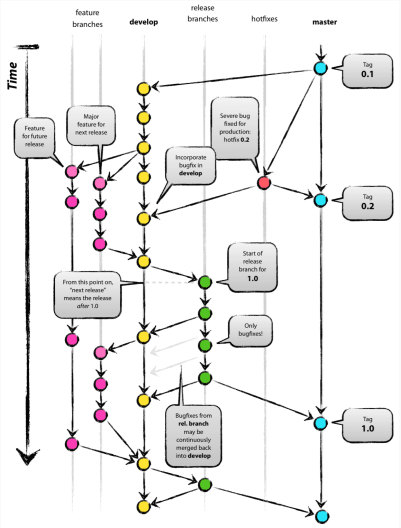
push

- Push your changes to the base repo
- Distribute your changes to others

Basic Commands - Status

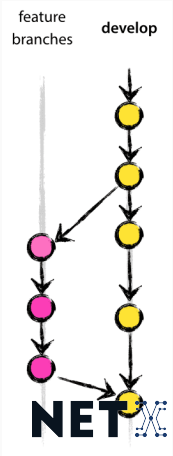
```
larsnielsen@darkstar-2 codes % git st
On branch development
Your branch is up to date with 'origin/development'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .lock-waf_darwin_build
        modified:   scripts/__init__.pyc
        modified:   scripts/waf/__init__.pyc
        modified:   scripts/waf/utils.pyc

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .waf-2.0.10-0b2f2e850da898fded177af41a17dd69/
        .waf-2.0.10-bf752372ae53c38f9478a0973bc78e0a/
        build/
        build_current
        resolve_symlinks/
        resolved_dependencies/
        src/codes/code_factory.cpp~
        src/codes/code_factory.hpp~
        waf_au_extensions/

no changes added to commit (use "git add" and/or "git commit -a")
```

- Separate your work
- Do not break others code
- Segment releases
- All repos are
- All repos has a master branch



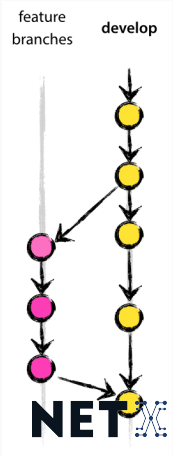
Working with branches

```
git checkout -b NEW_BRANCH_NAME BASE_BRANCH
...
git push -U origin NEW_BRANCH_NAME
```



- Create a branch
- work in it, *add*, *commit*, *push*
- Ensures minimal interference from others
- Ensures you don't screw up

```
git checkout BASE_BRANCH
git merge NEW_BRANCH_NAME
git branch -d NEW_BRANCH_NAME
```



- Create a branch
- work in it, *add*, *commit*, *push*
- Ensures minimal interference from others
- Ensures you don't screw up

Some time you may need to switch between branch.

```
git checkout BRANCH_NAME
```

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

Git flow is a branching model

- Semantic Versioning (not git) $\rightarrow$ BUT AWESOME
- It is used for tagging a specific commit
- Used for traceability

- A versioning scheme
- Has three types of release
  - Major
  - Minor
  - Hotfix

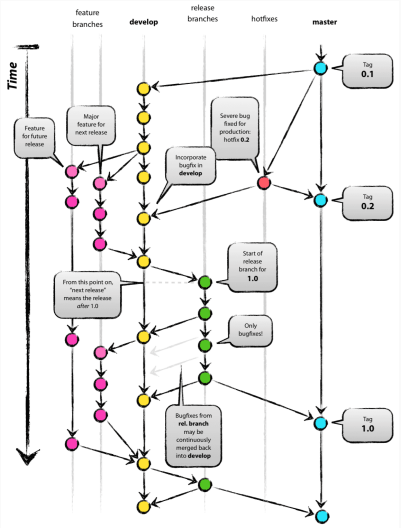
MAJOR.MINOR.HOTFIX

- A tag is a "form of a commit"
- We need to push tags

```
git tag -a VERSION -m "MSG"
git push --tag
```

- Easy to depend on externally
- Easy traceability
  - You can keep a log of your changes

- A *master* branch (stable)
- A *test* branch (stable)
- A *developer* branch (some what stable)
- *feature* branches (unstable)
- *release* branches (stable)



- A way to
  - Comment on merges
  - Reject a merge
  - Do code review
- Has different names
  - Pull Request
  - Merge request
  - Review request

- Magic little "programs"
- Fine grained control
- Can be annoying
- Placed in .git/hooks

- "Altering" commands
- You can change before and after
- Can be any scripting language you have an execution ENV for

```
    if [ "$allownonascii" != "true" ] &&
    test $(git diff --cached --name-only --
     diff-filter=A -z $against |
    LC_ALL=C tr -d '[ -~]\0' | wc -c) != 0
then
        cat <<\EOF
Error: ERROR MSG
EOF
        exit 1
fi
```

- Server Side
  - pre-receive
    - linting
    - user verification
- Client Side
  - pre-commit
    - linting

https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks

```
filename.extension
foldername/
entry*.extension
entry~
```

- Makes files ignored when added
- can be added with *-f*
- used for keeping the repo clean
- used for making the repo "safe"

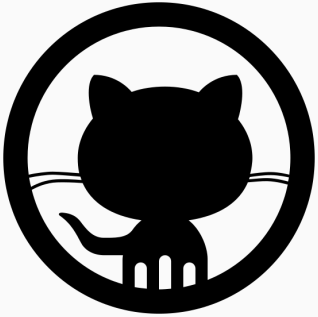
- File called *.gitconfig*
  - On macOS, BSD, and Linux in user root
- Allows:
  - configuring name and email
  - setting editor
  - aliasing
  - setting push and pull style
  - and more

```
[push]
        default = simple
[user]
        name = Lars Nielsen
        email = lnc13.lars@gmail.
    com
[alias]
        ...
        st = status
        ps = push
        pl = pull
        cb = checkout -b
        psu = push -u origin
        ...
[core]
        editor = emacs
```

AARHUS UNIVERSITY

```
git commit --interactive
```

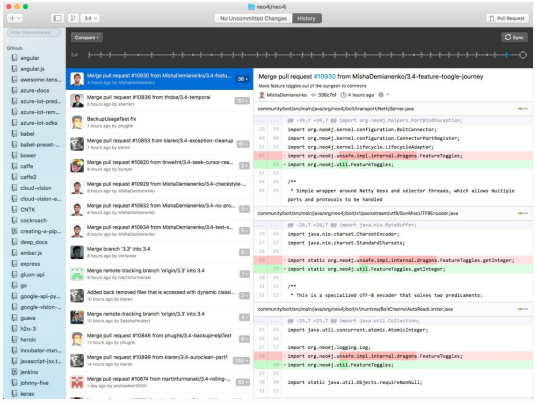
```
larsnielsen@darkstar-2 codes % git inter
           staged     unstaged path
  1:    unchanged        +2/-2 .lock-waf_darwin_build
  2:    unchanged       binary scripts/__init__.pyc
  3:    unchanged       binary scripts/waf/__init__.pyc
  4:    unchanged       binary scripts/waf/utils.pyc

*** Commands ***
  1: status       2: update       3: revert       4: add untracked
  5: patch        6: diff         7: quit         8: help
What now>
```



www.github.com

www.bitbucket.com

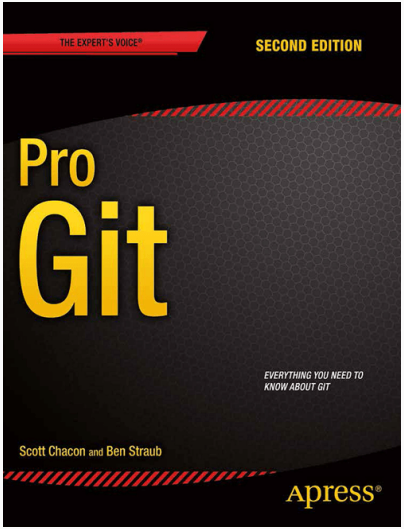
www.gitlab.com



https://desktop.github.com/

- Magit
- Emacs
- Marius Vollmer, Jonas Bernoulli, etc.

```
Head:     master README.md: use less bold text
Merge:    origin/master README.md: remove logo
Push:     origin/master README.md: remove logo
Tag:      2.8.0 (8)

Unstaged changes (2)
modified   README.md
modified   lisp/magit.el
@@ -1,4 +1,4 @@
-;;; magit.el --- A Git porcelain inside Emacs  -*- lexical-binding: t -*-
+;;; magit.el --- The Git porcelain inside Emacs and beyond  -*- lexical-bin>

 ;; Copyright (C) 2008-2016  The Magit Project Contributors
 ;;

Stashes (1)
stash@{0} cleanup readme

Unpulled from origin/master (4)
5aea99a * origin/master README.md: remove logo
04df2ff * magit-gpg-secret-key-hist: strip everything but the actual key
7fb3c23 * magit-stash-drop: log revision hash to process buffer too
29ff1ec * magit-mode-map: bind S-tab to magit-section-cycle-global

Unmerged into origin/master (1)
4f7ec47 * master README.md: use less bold text

 41: 0 UR-*magit: magit                    All   Magit Undo-Tree
```



Pro Git
Scott Chacon and Ben Straub
Free: `https://git-scm.com/book/en/v2`
Physical by Apress

Git Pocket Guide
Richard E. Silverman
O'REILLY