INTRODUCTION TO GIT

Trygve Aaberge

AGENDA

Introduction and history

How git works “under the hood”

Basic commands

Exercise 1 – Basics

Branching

Merging and rebasing

Exercise 2

Advanced topics

What is Git

- Distributed version control system
- The whole repository is on every machine
- Most commands are run locally
- Made to be fast and suited for multiple people working on the repo simultaneously and independently of each other

History

- Created by Linus Torvalds in 2005
- Made for development of the Linux kernel
- Replaced the proprietary BitKeeper
- Removal of the free use of BitKeeper spawned the creation of Git and Mercurial
- No existing free VCSs at the time met the demands of the Linux kernel

Some goals of Git

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

AGENDA

Introduction and history

How git works “under the hood”

Basic commands

Exercise 1 – Basics

Branching

Merging and rebasing

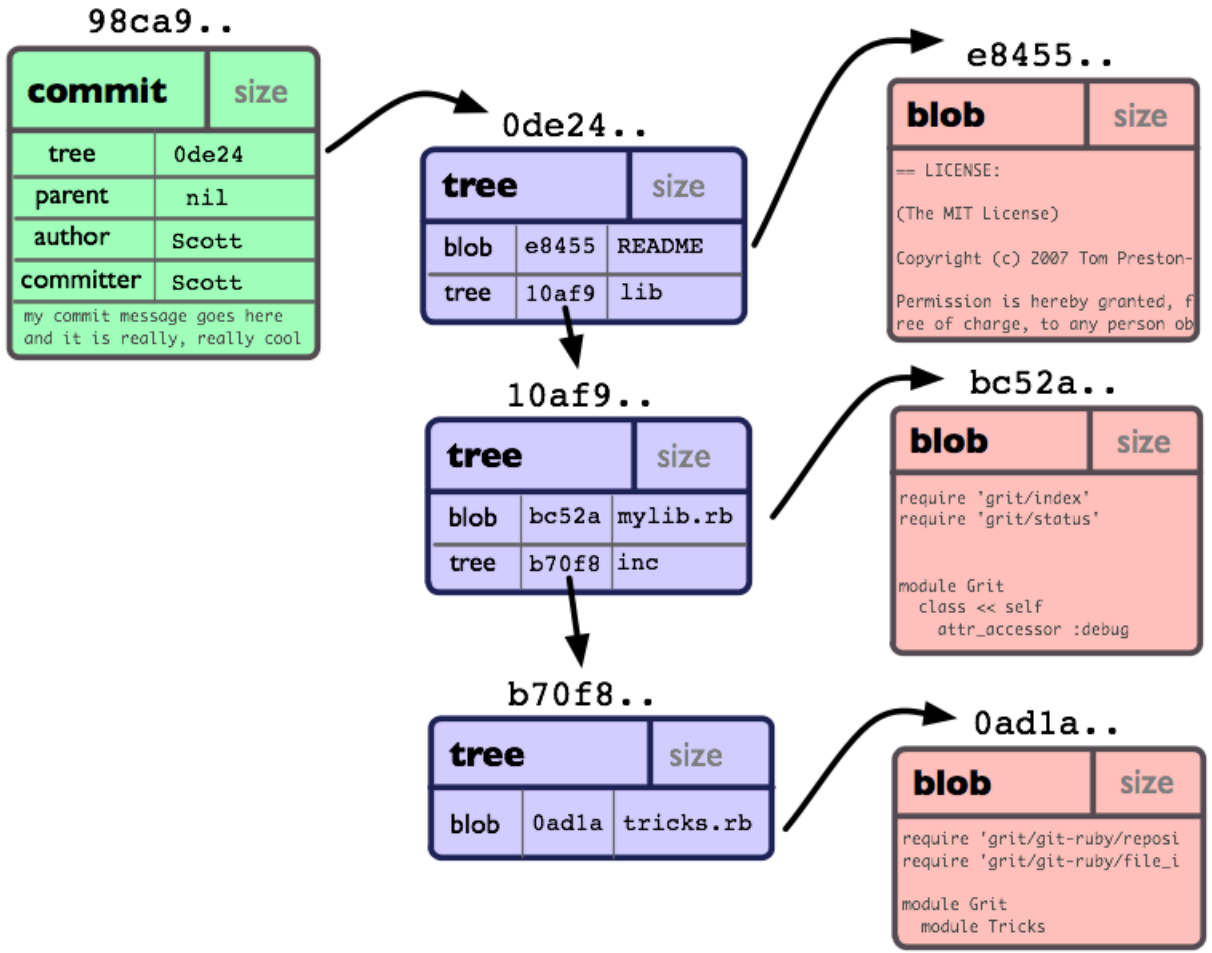
Exercise 2

Advanced topics

Stores snapshots

- Takes a hash of the content
- Stores the content as objects
- Any change makes a new object
- Tree-objects for file names and pointers
- Doesn't track renames
- Commit hash is computed from all content in the commit

Stores snapshots



Compressing snapshots

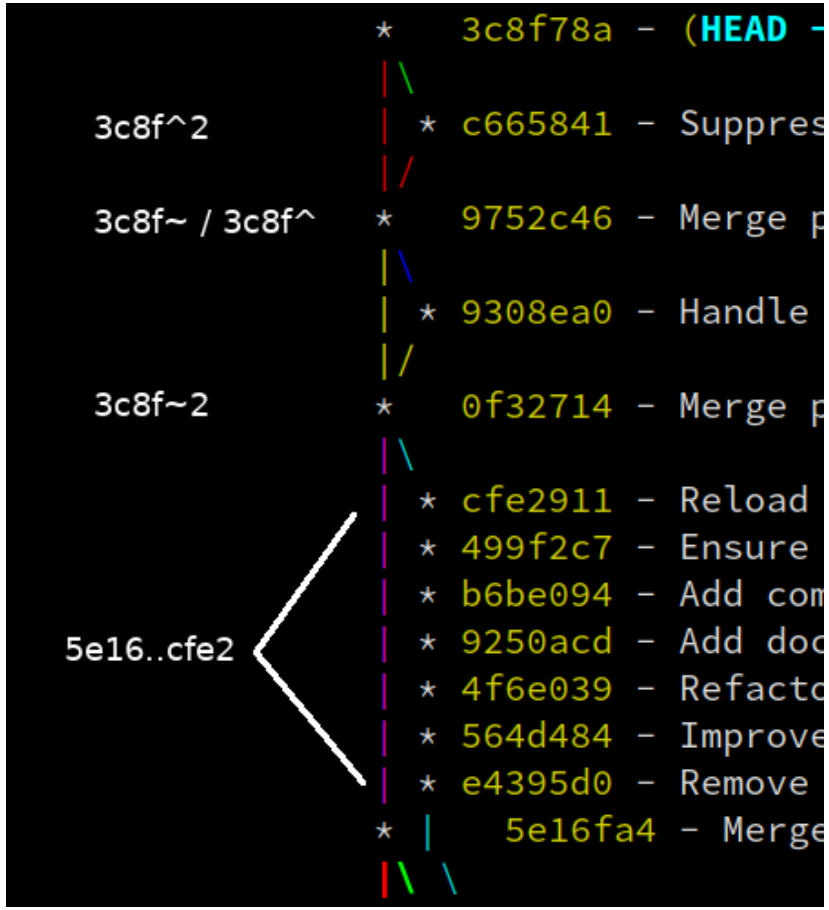
- Storing data as snapshots is fast
- Simple format
- However, makes for large amounts of data
- Compresses snapshots to deltas (the differences) when
  - There are many objects
  - When running `git gc`
  - Transferring over network

References

- Pointers to a commit
- Branches are references
- Remote branches
- `origin branch` vs `origin/branch`
- Tags, simple and annotated
- HEAD
- Detached HEAD
- FETCH_HEAD
  - GitHub specific: `git fetch origin pull/400/head`

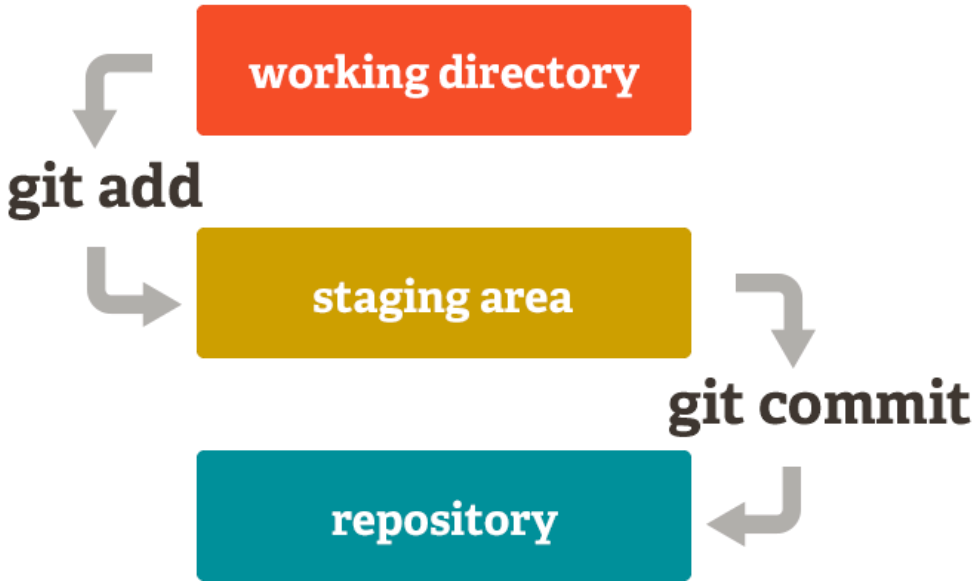
Revisions

- Any reference
- Any hash
- Other specifiers
- `ref^ vs ref~`
- `ref^-`
- `^ref og ref1..ref2`
- `man gitrevisions`



Staging area

- The next content to commit



AGENDA

- Introduction and history
- How git works “under the hood”
- **Basic commands**
- Exercise 1 – Basics
- Branching
- Merging and rebasing
- Exercise 2
- Advanced topics

Adding and committing

- `git add <file>`
- `git add -p`
- `git commit`
- `git status`

Branching

- `git branch`
- `git checkout -b branch-name [start-point]`
- `git branch -d / -D branch-name`

Working with remotes

- `git fetch`
- `git pull --rebase`
- `git push`
- `git push --set-upstream origin branch-name`
- `git push origin --delete branch`

Best practices

- Commit early, commit often
- One commit should only include related changes
- Include later fixups for that commit in it
  - Given that it hasn't been pushed yet
- Describe what the commit does and why it should be done

Best practices – Git recommends

- **Imperative form**
- **Wrap the lines to about 72 characters**
- **From Pro Git** (https://git-scm.com/book/en/v2/Distributed-Git-Contributing-to-a-Project)

```
Short (50 chars or less) summary of changes

More detailed explanatory text, if necessary.  Wrap it to
about 72 characters or so.  In some contexts, the first
line is treated as the subject of an email and the rest of
the text as the body.  The blank line separating the
summary from the body is critical (unless you omit the body
entirely); tools like rebase can get confused if you run
the two together.

Further paragraphs come after blank lines.

  - Bullet points are okay, too

  - Typically a hyphen or asterisk is used for the bullet,
    preceded by a single space, with blank lines in
    between, but conventions vary here
```

AGENDA

- Introduction and history
- How git works “under the hood”
- Basic commands
- **Exercise 1 – Basics**
- Branching
- Merging and rebasing
- Exercise 2
- Advanced topics

Exercise 1

https://github.com/ltera/git-fagkveld

Read the “Getting started” guide.

AGENDA

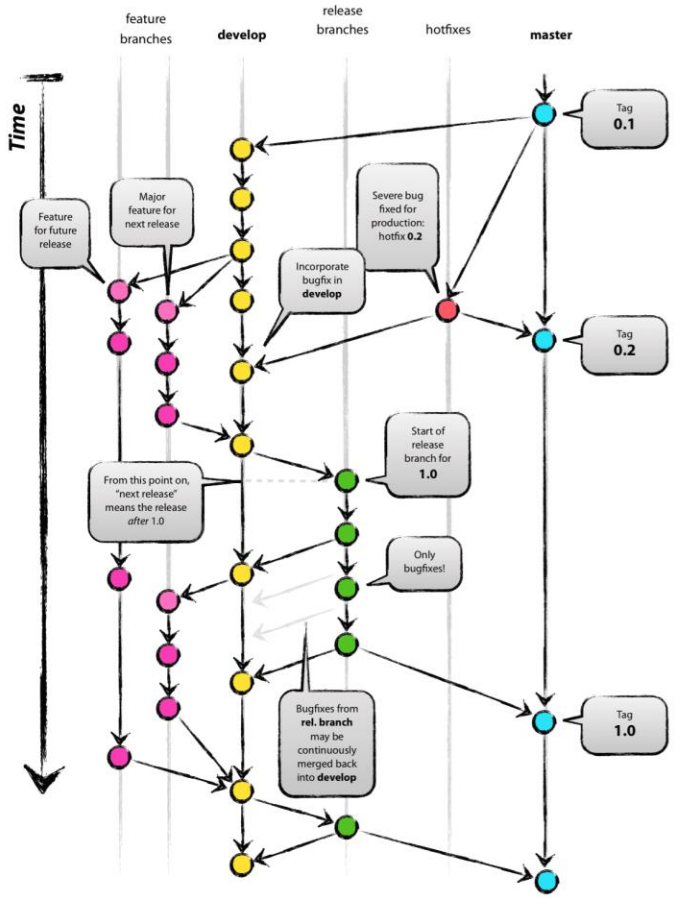
- Introduction and history
- How git works “under the hood”
- Basic commands
- Exercise 1 – Basics
- **Branching**
- Merging and rebasing
- Exercise 2
- Advanced topics

Branching

- Main branches
- Feature branches
- Long lived branches

Branching models

- Git flow
- Simpler alternatives

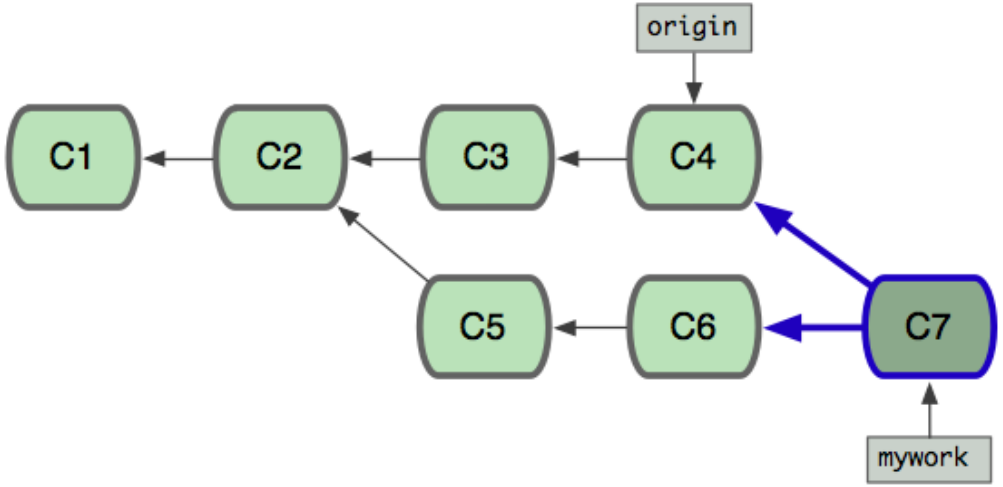


AGENDA

- Introduction and history
- How git works “under the hood”
- Basic commands
- Exercise 1 – Basics
- Branching
- **Merging and rebasing**
- Exercise 2
- Advanced topics

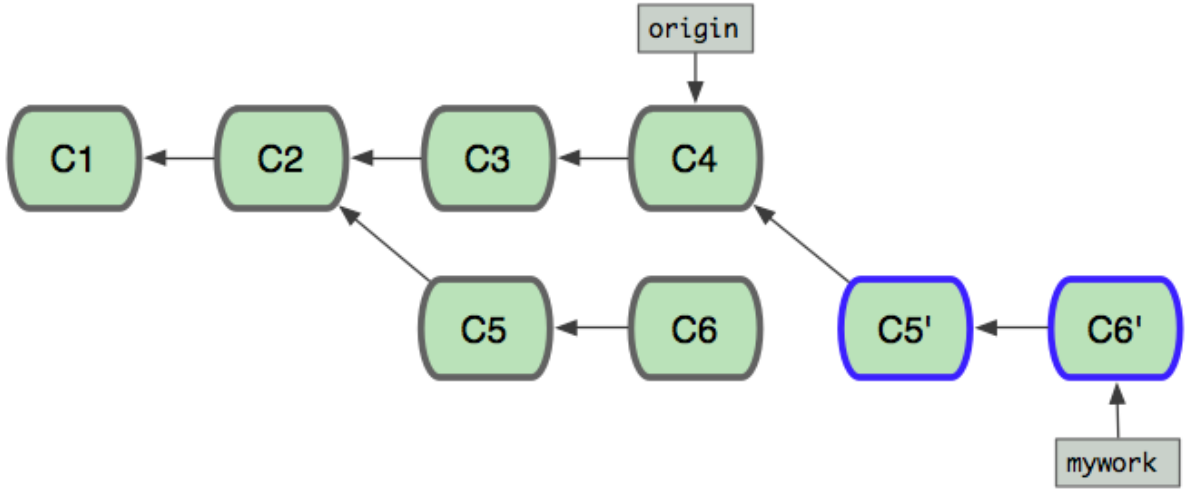
How does a merge work

- Fast-forward vs merge commit
- Parents of a merge commit
- Merge commits stores all of the content, as any other commit



What is a rebase

- Starts from a specific commit
- Reapplies the commits on top of that
- Allows you to make changes to earlier commits



When to use merge vs rebase

- Merge in feature branches
- Merge long lived branches
- Rebase when you want to change a branch
- Rebase when you want to make your branch up to date with another branch
- Individual opinions, some argue never to merge, some argue never to rebase

Disadvantages with merge/rebase to keep your branch up to date

- Disadvantages with rebase
  - You can't see the original commits
  - Individual commits may not build if you make a mistake
  - May need to resolve more conflicts
- Disadvantages with merge
  - Potentially harder to review
  - Harder to read the commit log
  - Harder to follow the original branches in the log

Merge conflicts

- Appears when two changes are done in the same place
- Git can't know what is correct, as it doesn't know semantics
- Resolve directly in file, or with a tool
- `mergetool`
- `merge.conflictstyle = diff3`
- You can ignore whitespace: -Xignore-space-change

Reverting merges

- Need to take special care
- You can revert a merge with `` `git revert -m1` ``
- If you merge the branch again, the changes won't be included again
- You can revert the revert instead

How to change earlier commits

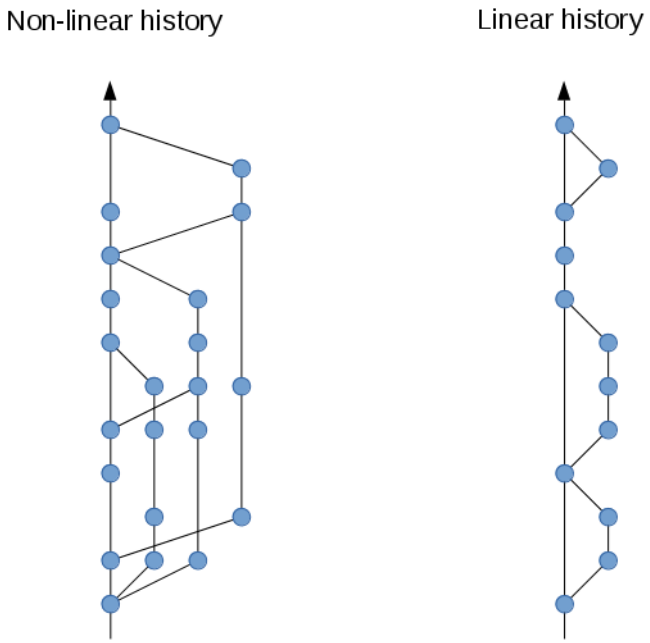
- `git commit --amend`
- `git commit --fixup + git rebase --autosquash`
- `git rebase -i`
- **Be careful when rebasing merge commits**
- **Removing commits**

Split a commit into two

- `git reset -p HEAD~`
- `git commit --amend`
- `git commit`

Make the commit history cleaner

- rebase will remove merge commits
- `git rebase <base> --onto <branch>`
- Join two commits into one



AGENDA

- Introduction and history
- How git works “under the hood”
- Basic commands
- Exercise 1 – Basics
- Branching
- Merging and rebasing
- **Exercise 2**
- Advanced topics

Exercise 2

- Amend a commit
- Fix an earlier commit
- Use interactive rebase
- Move a commit to another branch
- Merge back to master
- Resolve conflicts

AGENDA

- Introduction and history
- How git works “under the hood”
- Basic commands
- Exercise 1 – Basics
- Branching
- Merging and rebasing
- Exercise 2
- **Advanced topics**

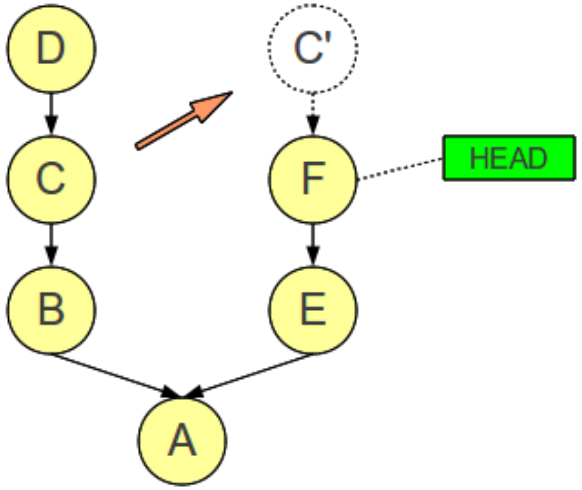
Reflog

- History of all your changes
- Default reflog, for HEAD
- Reflog for each branch

```
0f32714 HEAD@{0}: pull: Fast-forward
5e16fa4 HEAD@{1}: pull --prune: Fast-forward
3eff1de HEAD@{2}: pull: Fast-forward
d02bde0 HEAD@{3}: pull --prune: Fast-forward
5608e22 HEAD@{4}: checkout: moving from master-
f4e2ab2 HEAD@{5}: merge trygveaa/multiline-mess
1d4511d HEAD@{6}: merge FETCH_HEAD: Merge made
5608e22 HEAD@{7}: checkout: moving from master
5608e22 HEAD@{8}: pull --prune: Fast-forward
42b7746 HEAD@{9}: checkout: moving from master-
5c20f93 HEAD@{10}: merge FETCH_HEAD: Merge made
```

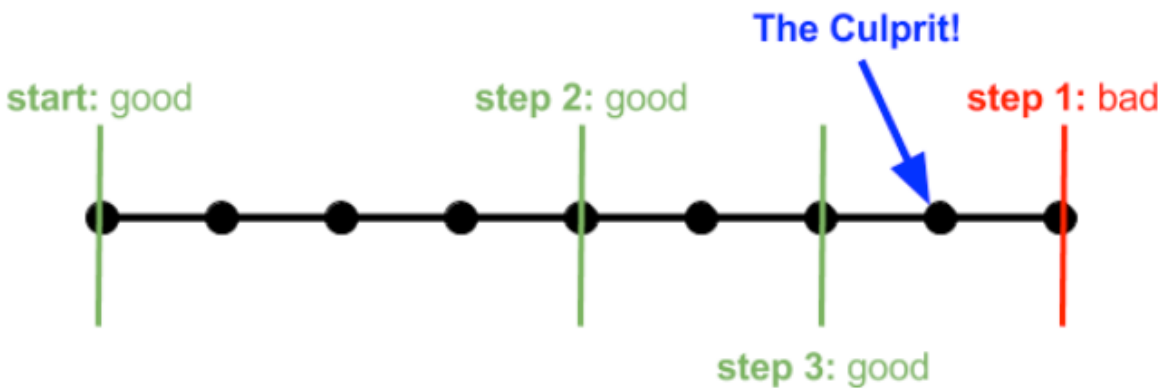
Cherry-pick

- Copies commits from one branch to another
- The commit hash changes



Bisect

- Used to figure out when bugs are introduced
- Does a binary search through the history



Worktrees

- Allows you to have multiple checkouts of a repo
- All the checkouts shares the git directory
  - I.e. commits, references, stash, etc.
- Not allowed to check out a branch multiple places simultaneously

Thanks!

Questions?