git in a Nutshell

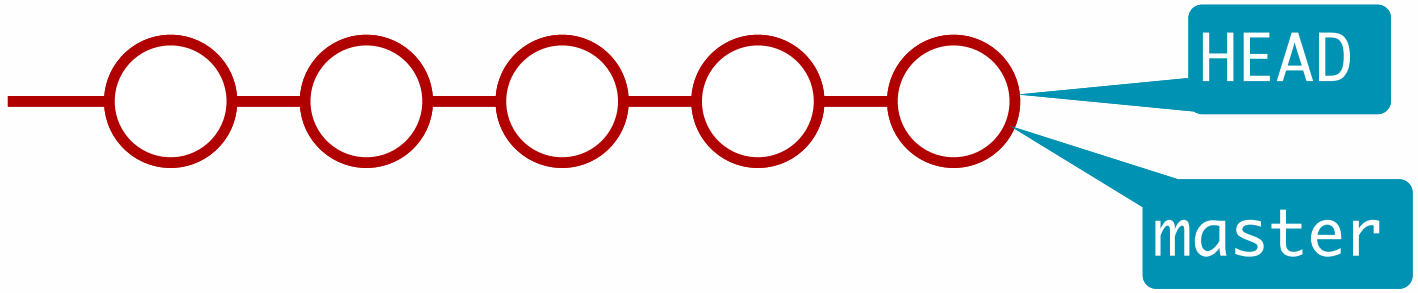
Software Systems — Design — L5T3

Dr. Vadim Zaytsev aka @grammarware, November 2020

UNIVERSITY OF TWENTE.
Formal Methods & Tools

Source Code Management

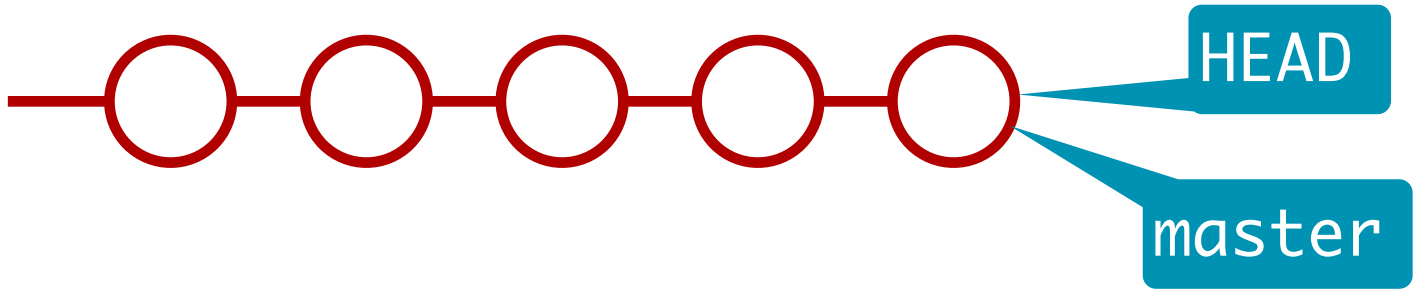
git clone



remote

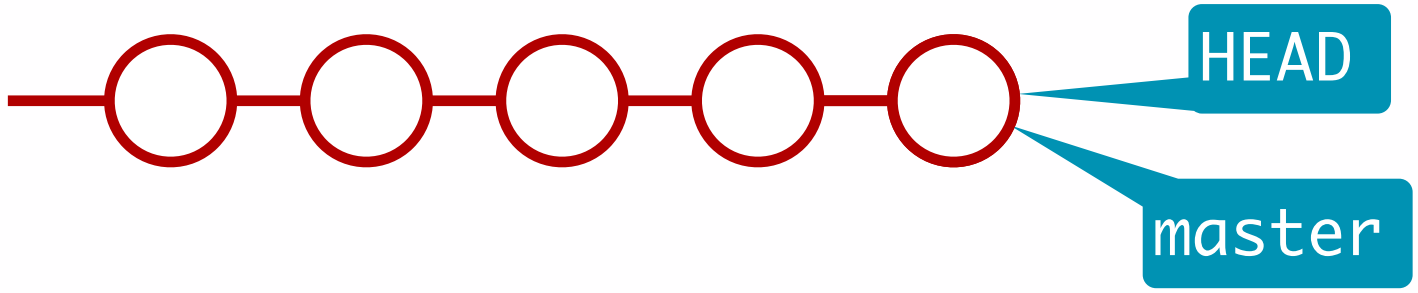
local

git commit

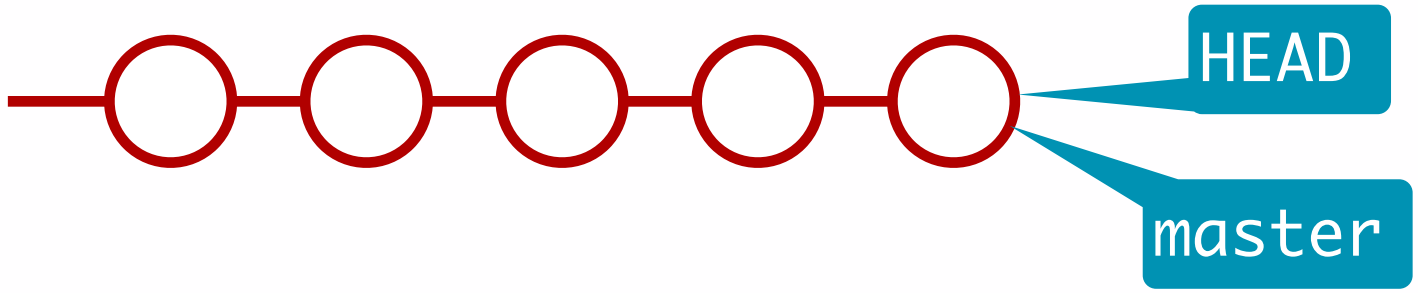


remote

local

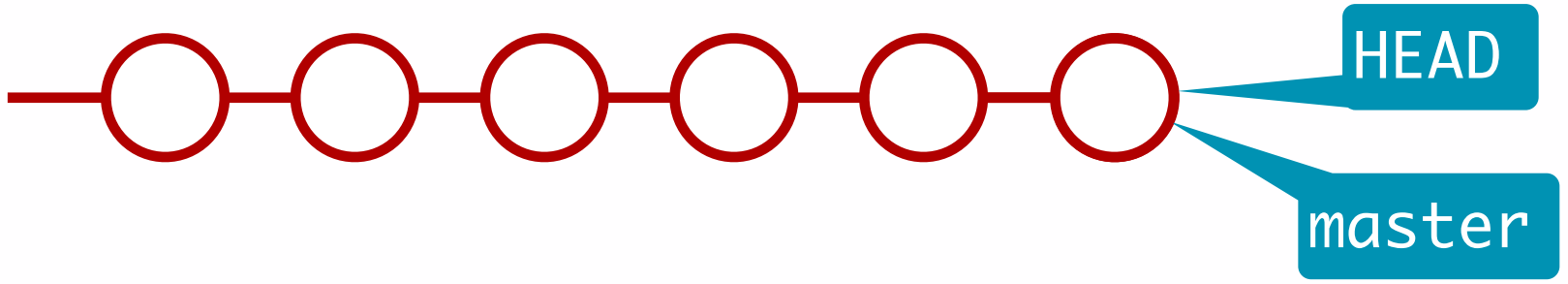


git push

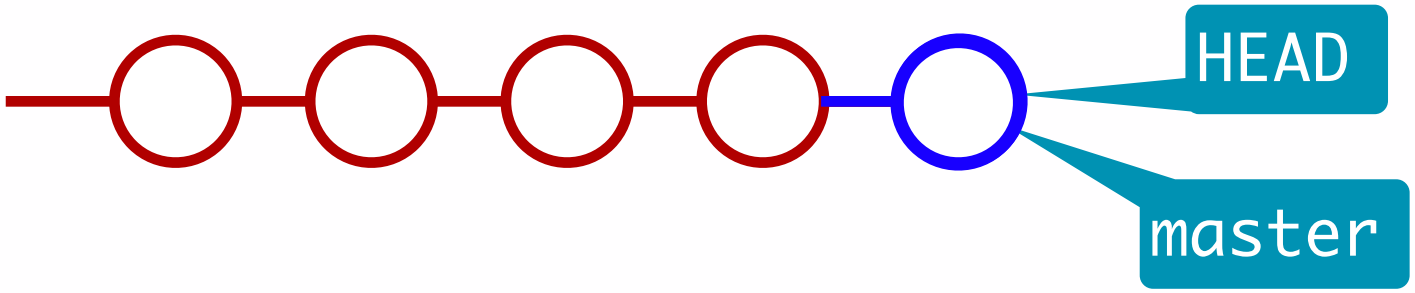


remote

local

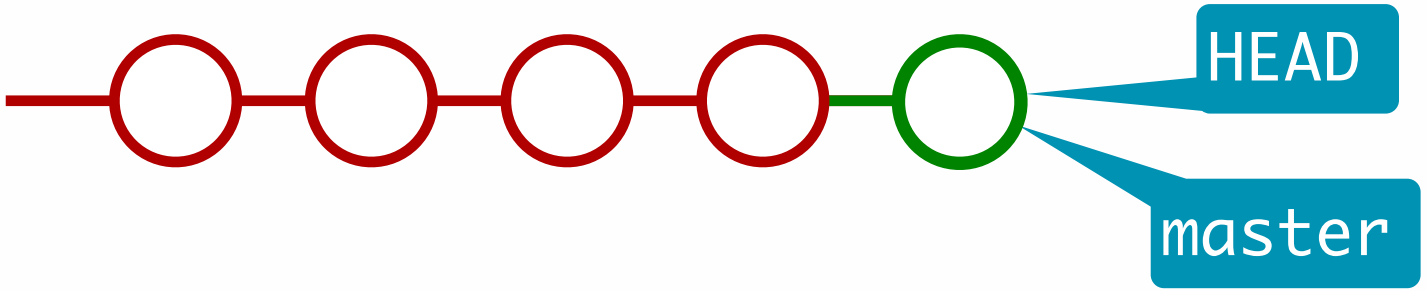


git fetch

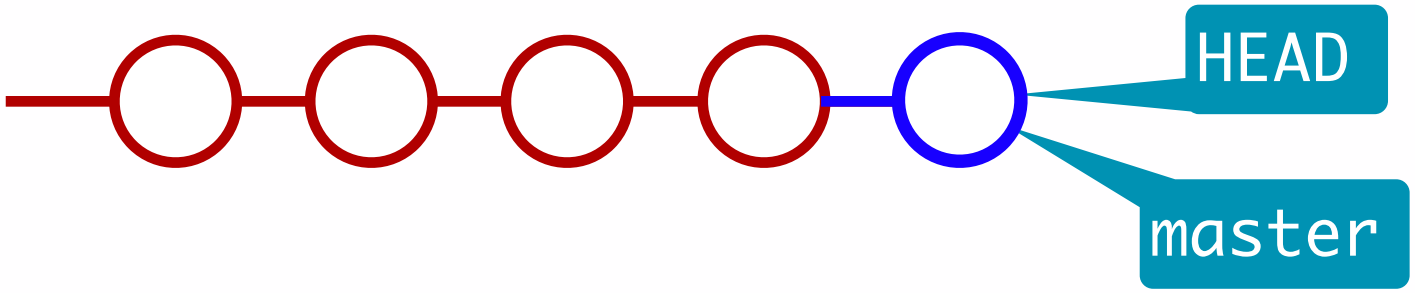


remote

local

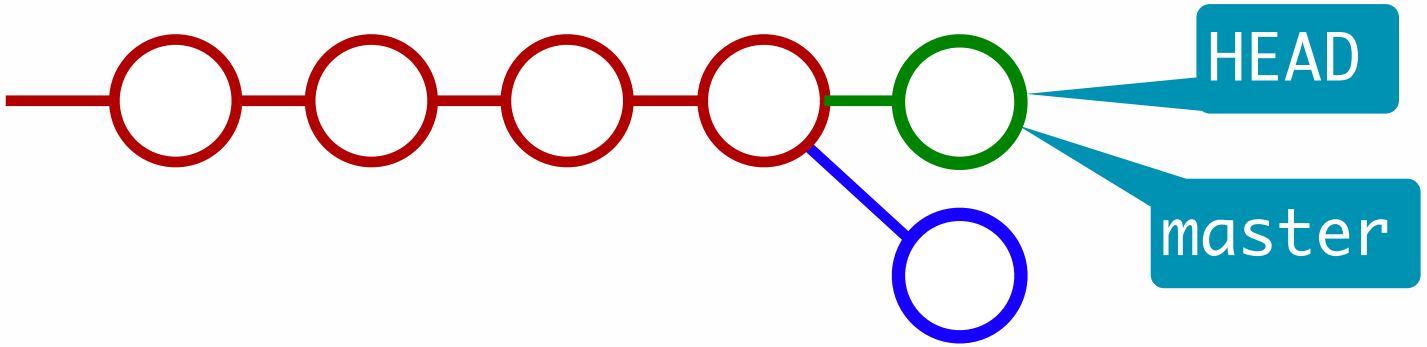


git rebase

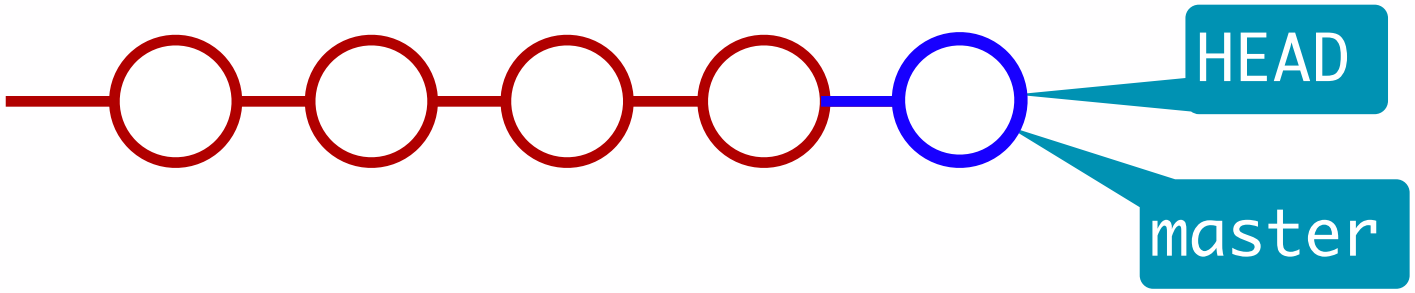


remote

local

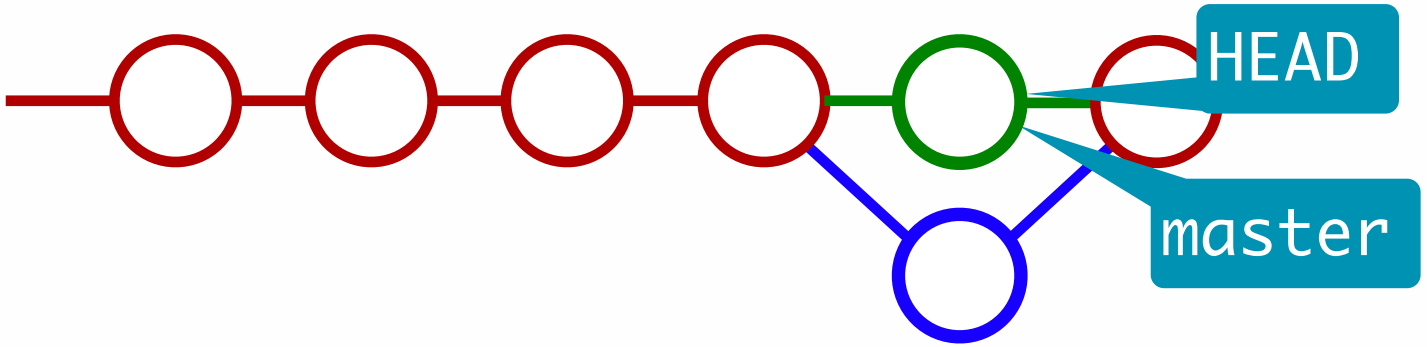


git merge

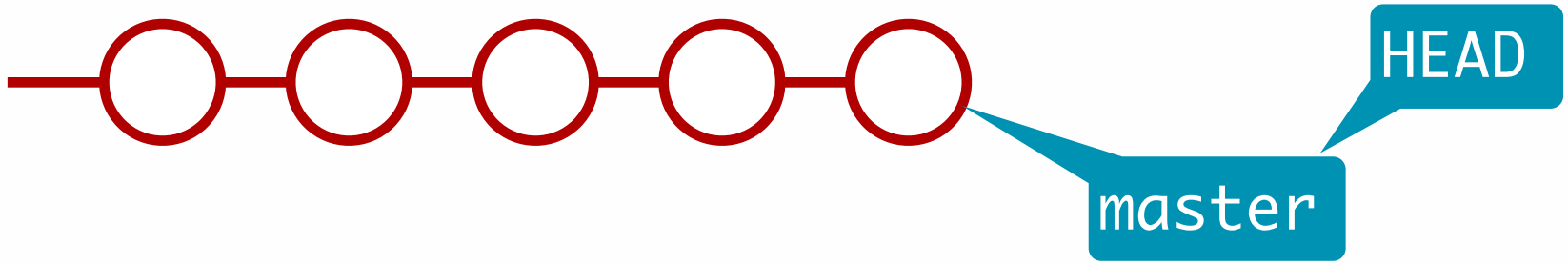


remote

local

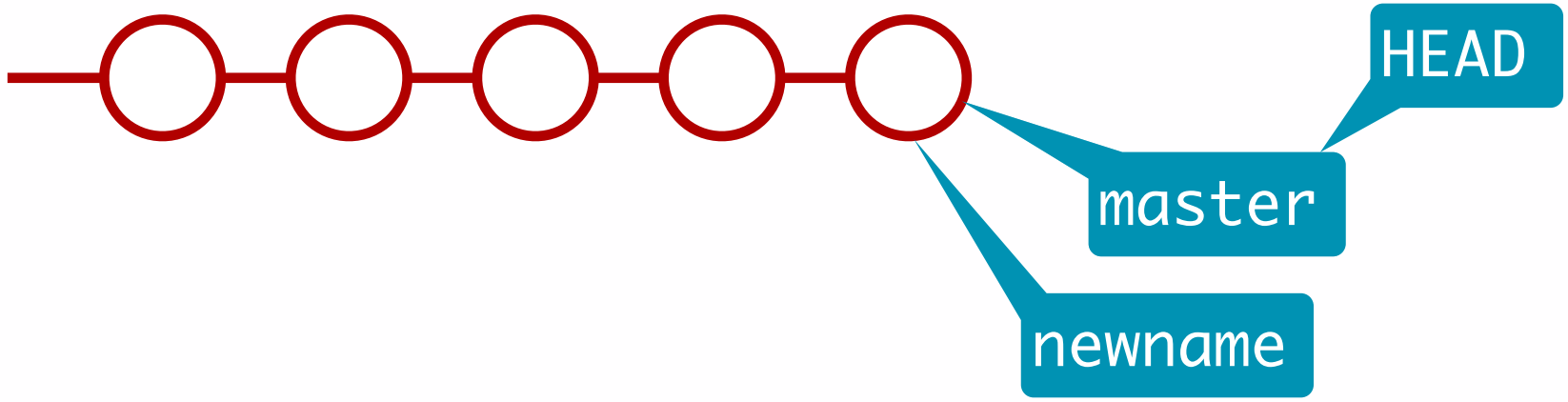


git branch

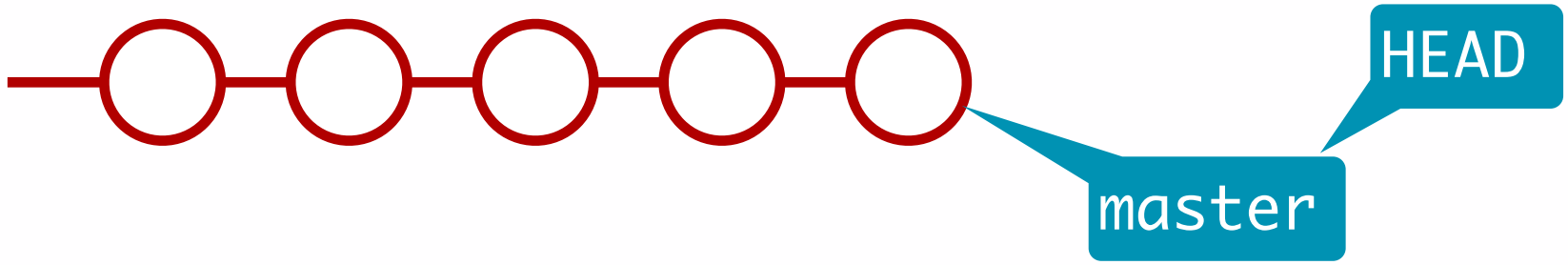


remote

local

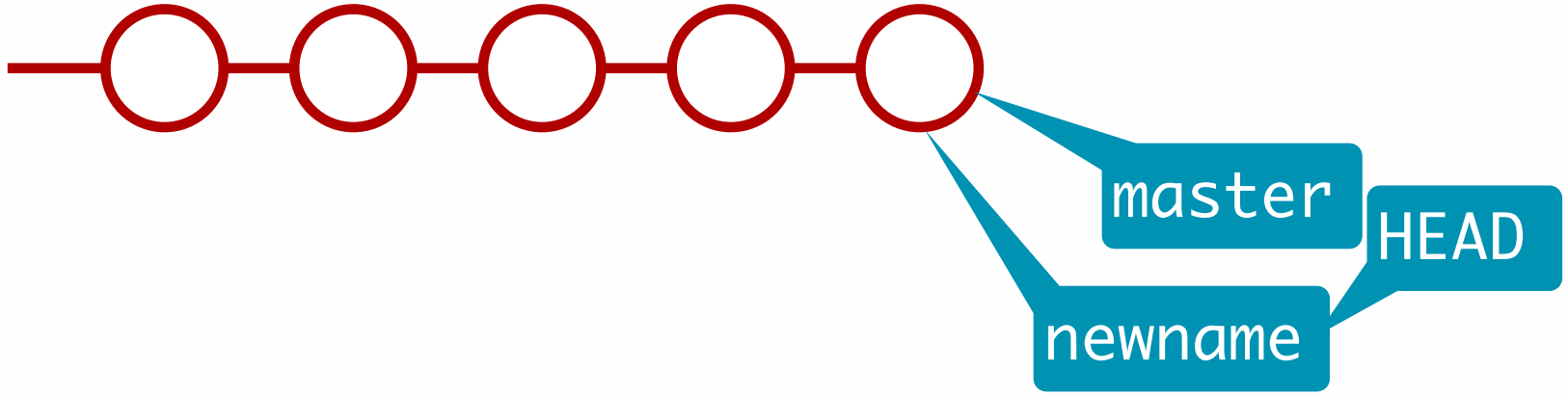


git checkout <branch>

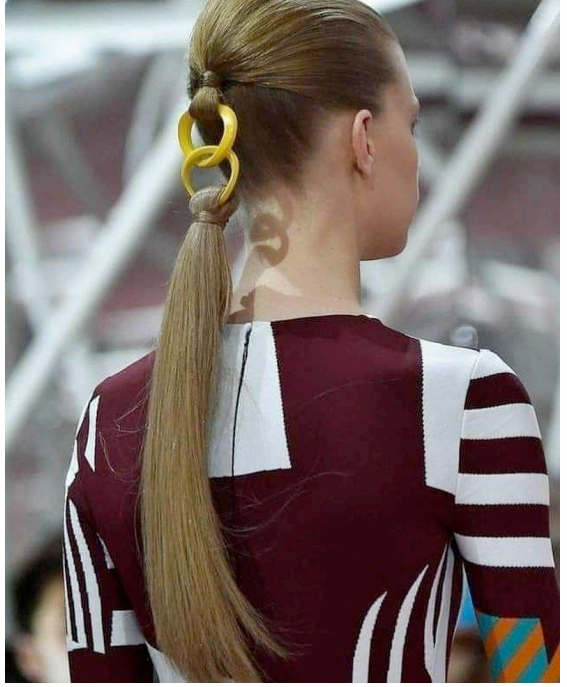


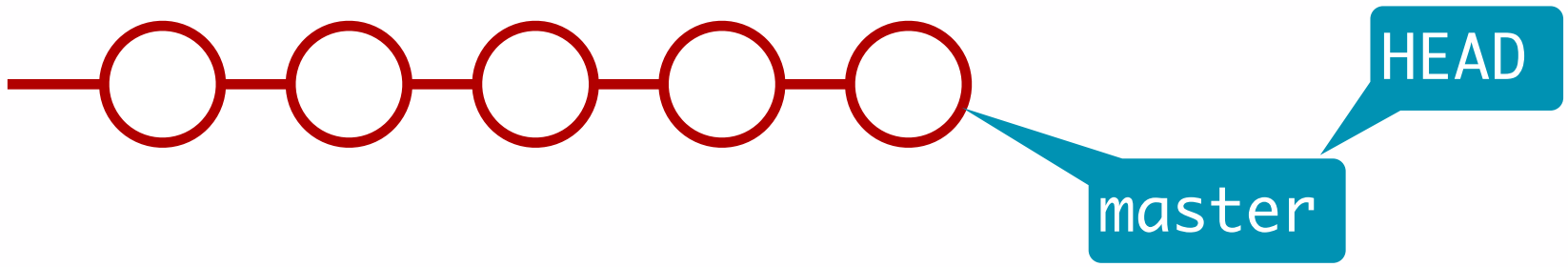
remote

local



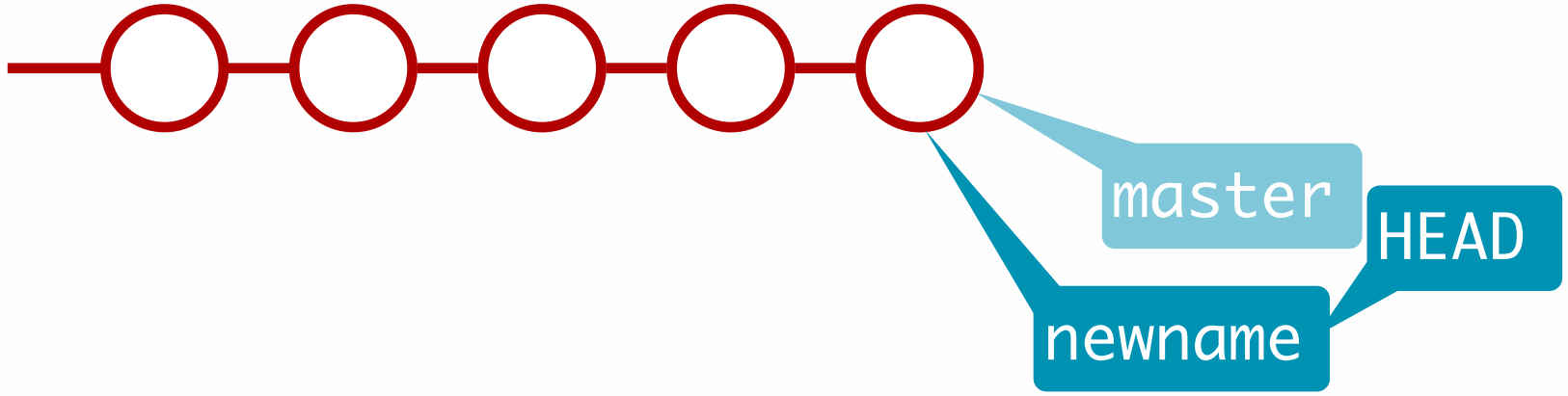
git checkout <commit>





remote

local



Remaining Jargon

- `git log` show last commits
- `git blame` show the last change per line
- `git pull` = fetch + merge
- `git revert` undoes one commit
- `git cherry-pick` applies one commit
- `git show` show commit details

Guidelines

- One commit – one purpose
- Commit as often as possible
- Push often to prevent merge conflicts
- Confine inconsistent states to branches
- Write commit messages
- Do not alter history
- Make rules and follow them

Best practices for using Git; *Git Best Practices*; etc

Guidelines on Commit Messages

- Make it findable
- Use the imperative mood
- Explain what and why (not how)
- Separate subject from body
- Keep subjects under 50 chars
- Capitalise the subject line
- Do not add dots

Chris Beams, *How to Write a Git Commit Message*, 2014

Conclusion

- git is big and full of horrors
- Most of the time you will do
  - add, commit, pull, push
- Make full use of GUIs
  - Eclipse
  - GitKraken, gitx, Git Extensions, …
- Find the process that suits you
  - and follow it

Topics/Slides Disclaimer

- Good ✅
  - watch before Q&A
  - embrace reality
  - try out at labs
  - ask for feedback
  - apply to project
  - dig deeper
  - recall from slides
- Bad ❌
  - slides over videos
  - assumptions
  - blanks
  - timing