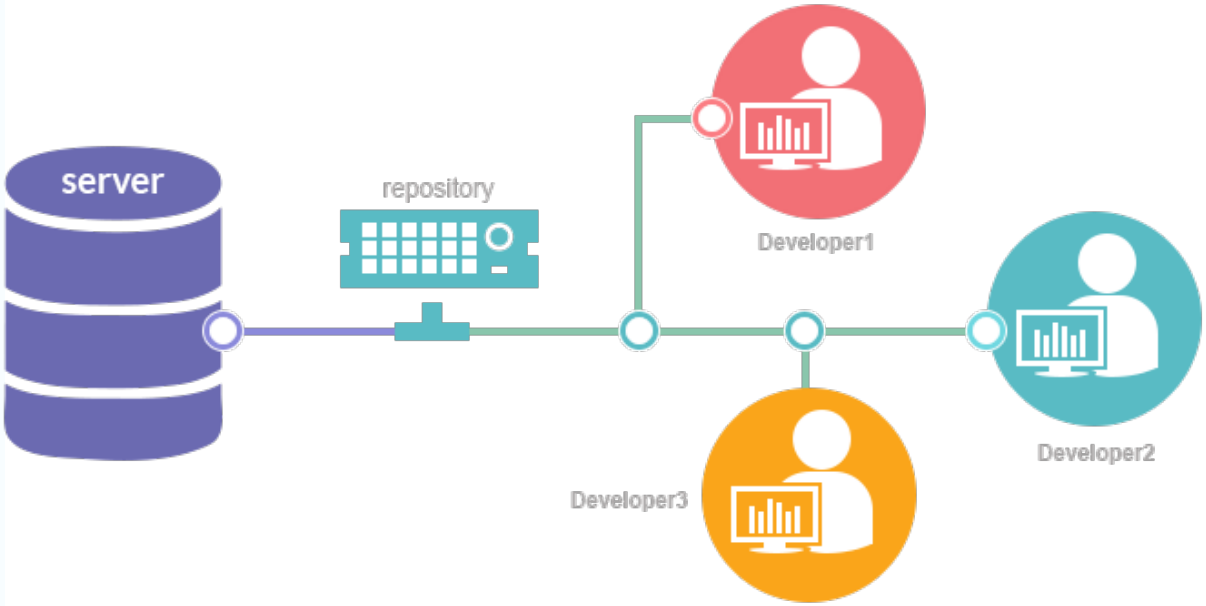
Dainius Masiliūnas
r-users

2018

Version Control Systems (VCS)

- Code development requires:
  - Collaboration between multiple people at once
  - Backing up files
  - Keeping track of file history (in case something breaks)




PERFORCE
Version everything.


monotone


Microsoft
Visual
SourceSafe 6.0


CVS


Bazaar


BitKeeper
Scalable Version Control


SUBVERSION


mercurial


PERFORCE
monotone
git
BitKeeper
Scalable Version Control
SUBVERSION
mercurial

History of git

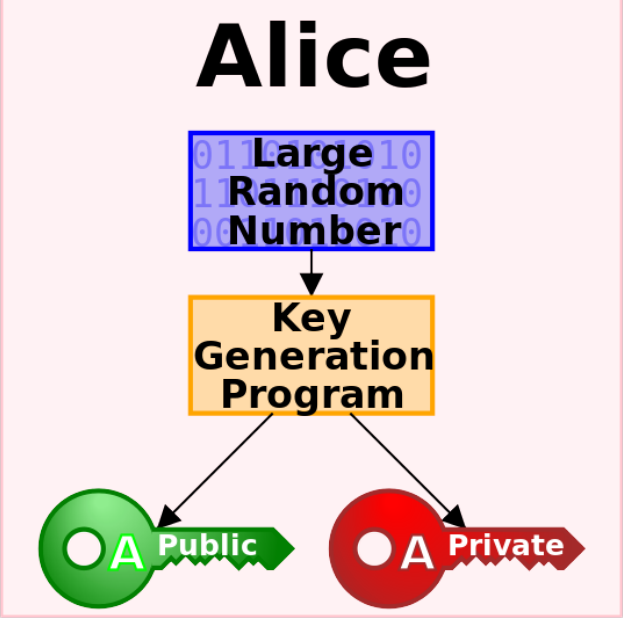
- Git: Free and open-source version control system
- Made by Linus Torvalds, creator of Linux, for Linux
  - 61,308 files, 24,163,085 lines, 723,659 commits, 16,914 developers (2018-01-02)
- BitKeeper: used to have a freeware version, but stopped in 2005
- Git created as a replacement, “inspired” by CVS
- Named after Linus: *Git [git] (British Slang) a foolish or contemptible person.*
- Overtook all other VCS

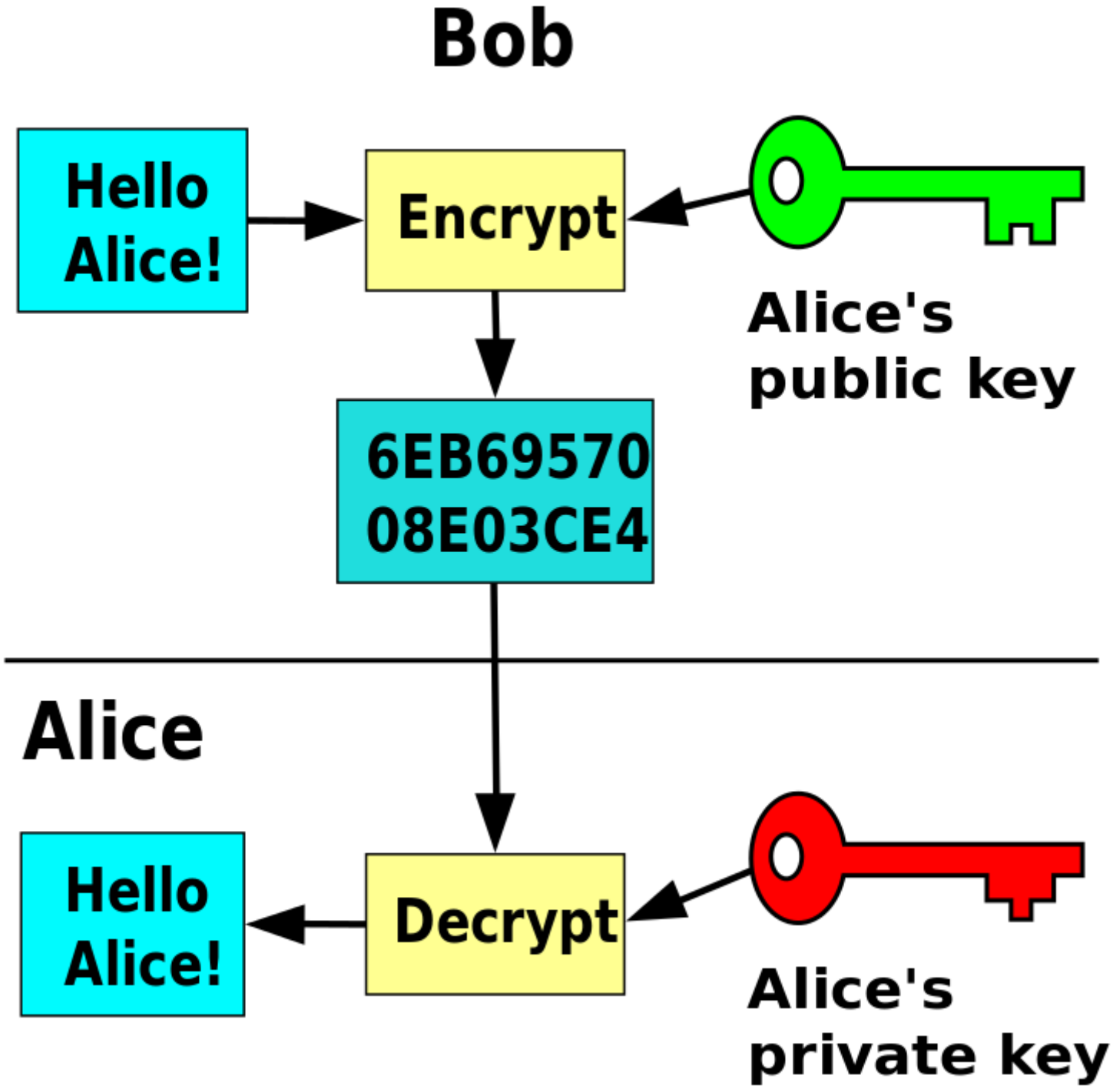
Why Git?

- Distributed
- Fast
- Space-efficient (for text)
- Cross-platform
- Flexible
  - Forking, branching, merging
  - Advanced: git bisect, git blame
  - Commit early, commit often (but not too often)

SSH keys

- No security model by itself: delegates it to SSH
- Uses SSH (RSA) key pairs (private and public) for better security (and convenience!)
  - Public key identifies and grants access to owner of the private key
  - Private key is never actually sent over network, each **machine** has a unique key
  - Private key can be encrypted with a passphrase to guard against theft/viruses
- Also used by SSH for connecting to remote computers




Bob
Hello Alice!
Encrypt
Alice's public key
6EB69570 08E03CE4
Alice
Hello Alice!
Decrypt
Alice's private key

Git hosting websites

- **GitHub**: Microsoft's proprietary git hosting service (paid for private repositories)
  - Very popular: 24 million users, 67 million repositories, e.g. Linux kernel, CRAN and individual R packages, Unreal Engine 4 (statistics: https://octoverse.github.com)
  - Student Pack available for students
- **GitLab**: Open-core git hosting service (free private repositories, Enterprise Edition has more features)
  - Second most popular, anyone can host their own instance
  - WUR already does: https://git.wur.nl

GitLab

Microsoft GitHub

GitLab/GitHub features

- Forking repositories
- Collaboration (on one or more repositories)
- Bug and task tracking
- Documentation
- Simple websites, e.g. Geoscripting course
  - Hence http://geoscripting-wur.github.io/
  - Source: https://github.com/GeoScripting-WUR
- Example of a large project: https://github.com/nextcloud/server

Uses for Git{,Hub,Lab} from personal experience

- 2007: I started to use Linux (SUSE 10)
- 2008: Linux as my primary OS
- 2009: GitLab repository for sharing code and collaboration (game mods); 2017: a team of 4 people on GitHub
- 2012: Git bisect for debugging the Linux kernel
- 2014: Package maintainer in Gentoo Linux
- 2016: Contributed to Unreal Engine 4 on GitHub; ACT: GitHub issues for managing tasks in a team (SCRUM)
- 2017: Master theses (text + code) on GitHub, reported a number of bugs in R packages (fixed in ~1 month)

Basic workflow

- Create a git repository in Git{Hub,Lab}, add at least one file
- **`clone`** the repository to your PC
- Put in files you want to keep track of (text!), **`add`** them to the repository
- **`commit`** (checkpoint on your PC)
- **`push`** (send changes to server)
- **`pull`** (fetch changes from server + merge them to your local repository); beware of merge conflicts

Git user interfaces

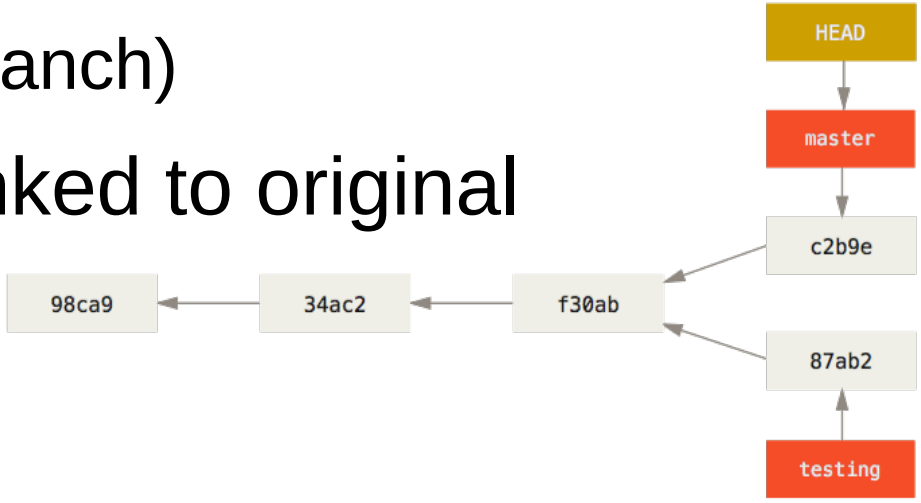
- `git` (use from a terminal)
- `git gui` (comes with git, can start from a terminal)
  - Really useful for SSH key management!
- RStudio has an integrated client
  - But you don’t want to use RStudio for Python etc.!
- Third party: https://git-scm.com/downloads/guis/

R and git integration

- Packages: structure your code as if it was a package, and it will be a package
- `library(devtools)` provides `install_git{,hub}(url)`
- Example: https://github.com/appelmar/bfast
  - Fork: https://github.com/GreatEmerald/bfast
- Example with Roxygen: https://github.com/loicdtx/bfastSpatial
- Git{Hub,Lab} can render Jupyter notebooks, e.g.: https://github.com/GreatEmerald/master-classification/blob/master/examples/Demo.ipynb

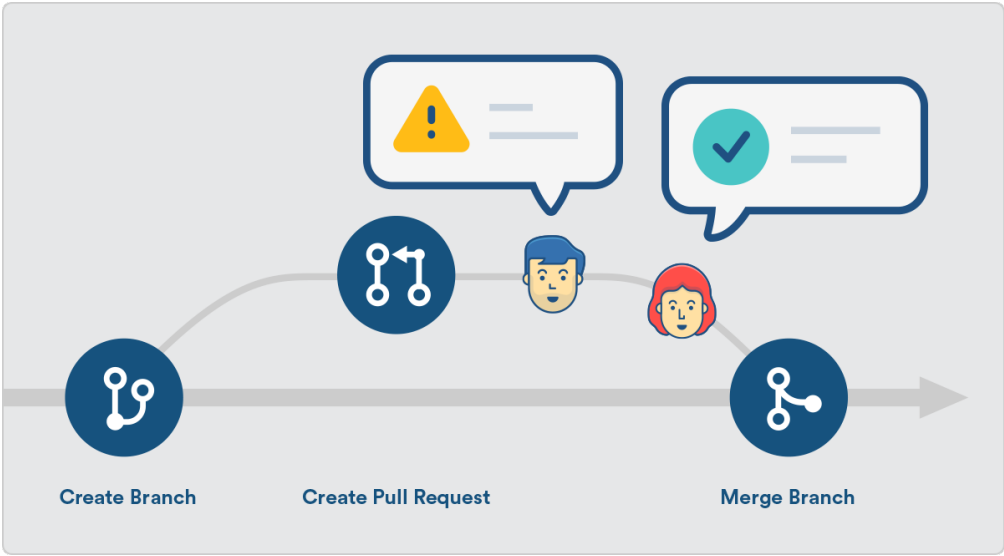
Branching and forking

- Branch: a series of versions of your files
- Branches (usually) have a common ancestor
- You use branches without even knowing: your local repository is a branch compared to remote
- Branches can be:
  - Merged into one another
  - Stay separate (e.g. devel branch)
- Fork: repository copy, unlinked to original



Pull requests

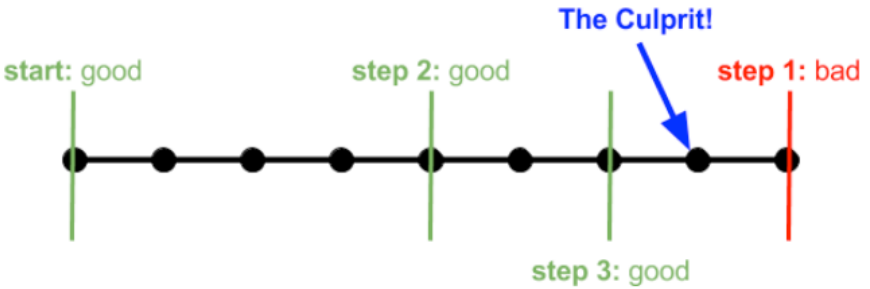
- You found a bug!
- You found a way to fix the bug! But it’s not your code. Solution:
  - Fork the repository (you become “downstream”)
  - Fix the bug
  - Submit a pull request: ask to merge your changes into “upstream”



- Upstream is happy to accept your changes
- Bug is fixed for everyone in the world!

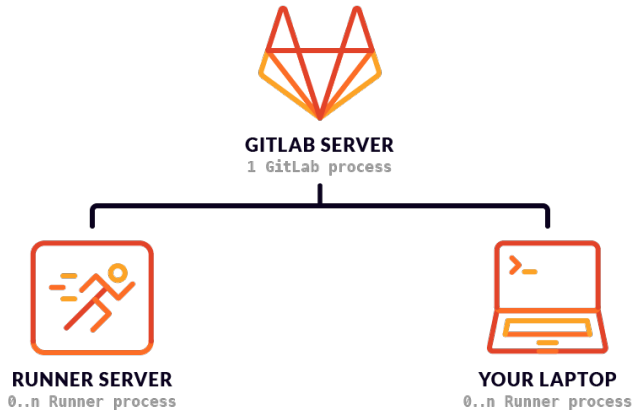
git bisect

- You implemented a feature at some point, it worked well
- At some point it broke/there was a bug!
- What caused it to break?
- If you know the line that broke it: `git blame`! If not, bisect.
- Bisect is a binary search:
  - `git bisect good <commit>`
  - `git bisect bad <commit>`
- Test each commit until you find the culprit!
- Don’t commit broken code. In case you did: `git bisect skip`

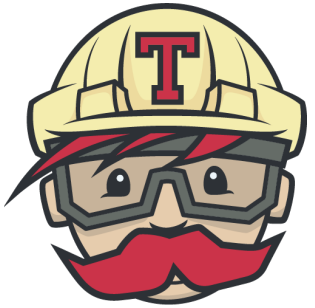


CI (Continuous Integration)

- You can break your code without realising it
  - Pull requests from outsiders
  - Commits that do small changes that end up breaking everything
- Could test every single commit before accepting it, but that's a lot of work
- Solution: set up a CI:
  - For each new commit, creates a VM
  - Runs your defined test cases
  - Reports whether it passed or failed



- Creative uses for CI: linting, binary file checksumming, plagiarism detection



Thanks for listening

Questions?