Git Essentials

Bartosz Majsak, Thomas Hug

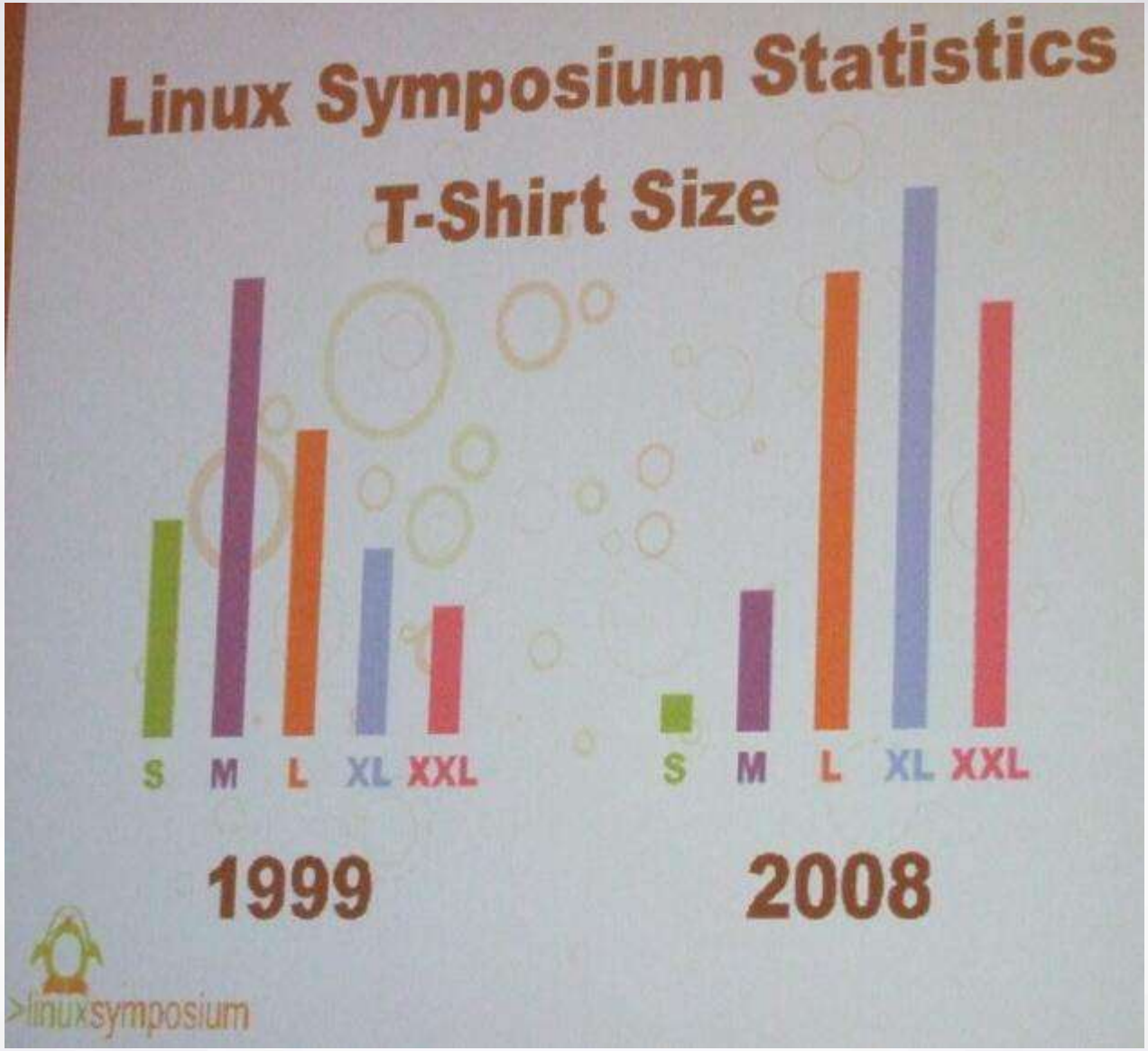
About Us

Cambridge Technology Partners

- Bartosz Majsak
  - Java Developer by day
  - Open source junkie by night (Arquillian core team member)
  - Conference speaker by passion (Devoxx, Jazoon ...)

- Thomas Hug
  - With Cambridge Technology Partners since 2002
  - Java Developer, TTL, Solution Architect
  - Apache Committer, OSS contributor and aficionado

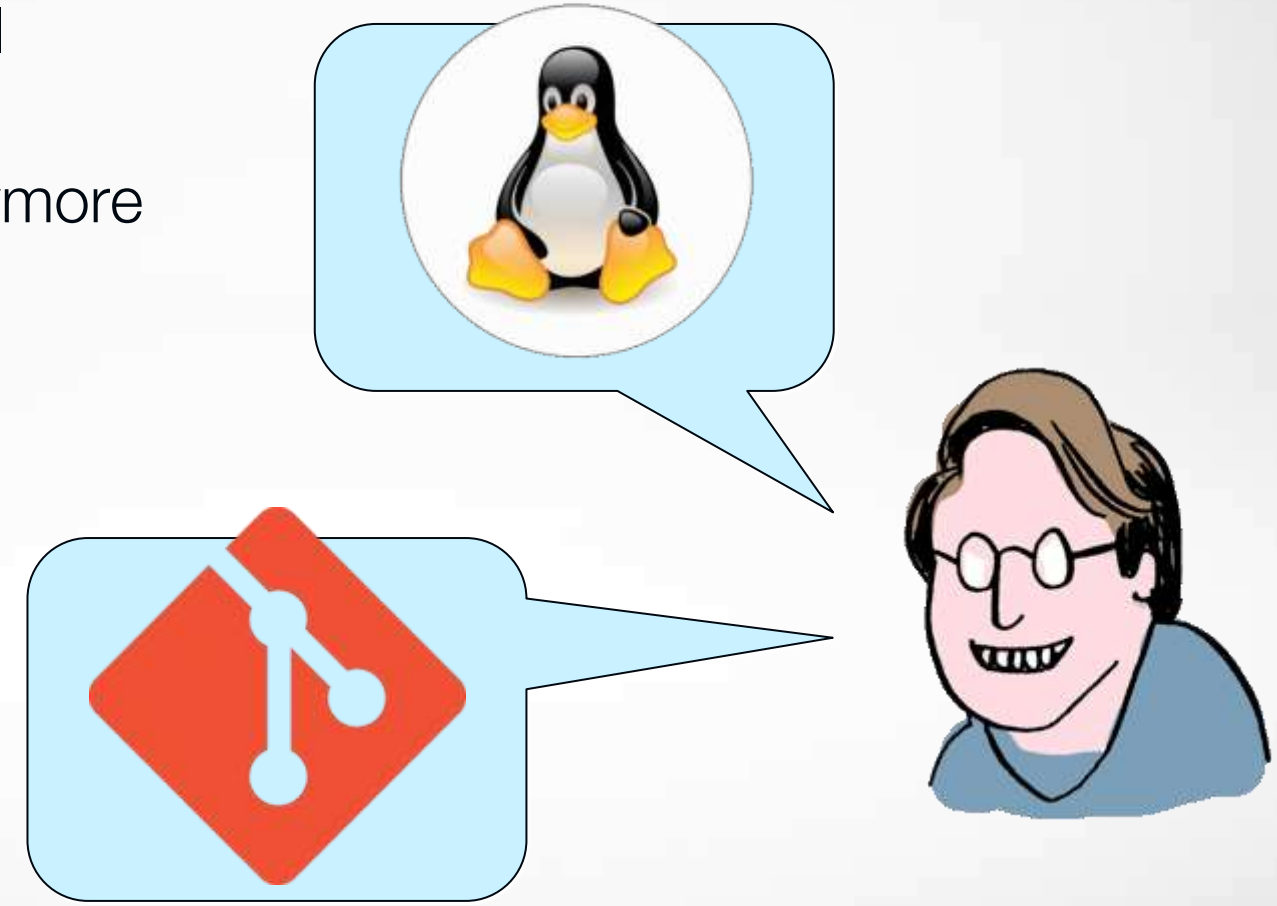
Why do we recommend Linux?



Git a British slang term meaning a contemptible person, a bastard.

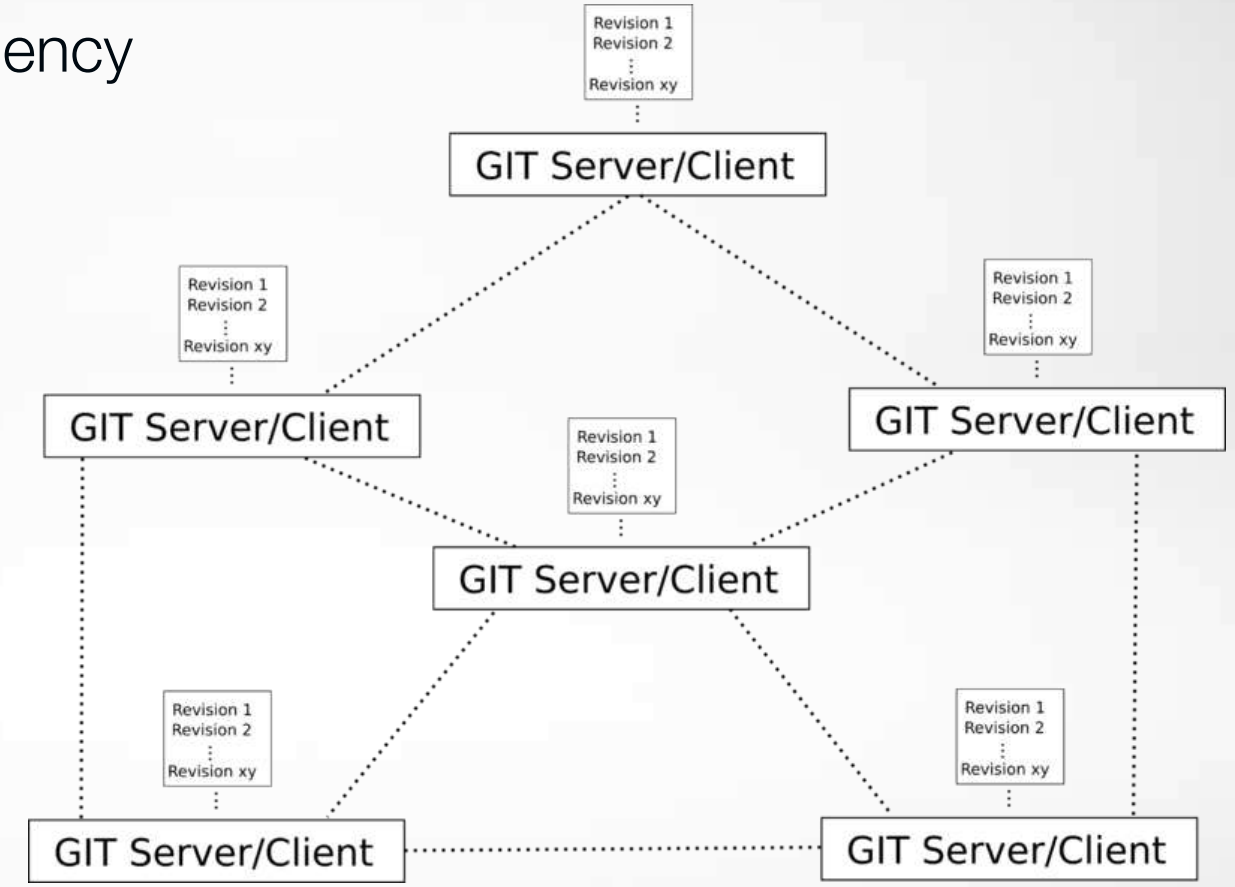
Git History

- Founded 2005 as a replacement of BitKeeper
- VCS of Linux Kernel
- …not just Linux anymore



Git Concepts

- No Central Server – Distributed VCS
- Performance and Efficiency
- Robustness



Disclaimer when we say **repository** we actually mean **local repository (no network connectivity)**

Cambridge Technology Partners

Installing and Configuring Git

Installation

- msysgit
- cygwin
- Atlassian SourceTree

- XCode
- Homebrew
- MacPorts

- Package Manager

Cambridge Technology Partners

Playground

Objectives: getting familiar with essential commands and making your life easier

- `touch`
- `cat / less`
- `mkdir`
- `ls`
- `tree`
- `cp / rm / mv`
- `nano`
- `history / ctrl+shift+r`

User Identity

Cambridge Technology Partners

- Your contact details

```
$ git config --global user.name “Bruce Wayne”
$ git config --global user.email “batman@gotham.com”

$ less ~/.gitconfig
```

- SSH Key generation

```
$ ssh-keygen -t *dsa -C batman@gotham.com
```

**Using SHA-2 underneath. Approved by NSA*

Presets

Cambridge Technology Partners

- Color output

```
$ git config --global color.ui auto
```

- **Aliases**. Useful for stuff impossible to remember…

```
$ git config --global alias.showlog "log --color --
graph --pretty=format:'%Cred%h%Creset -
%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold
blue)<%an>%Creset' --abbrev-commit"
```

Three levels of configuration:

--local (default, per repo) --global (per user) --system (machine)

Reference Material

Cambridge Technology Partners

- Git References
  - http://git-scm.com/ - official Git Home
  - http://git-scm.com/book - Pro Git (Apress) online version
  - http://git-scm.com/docs - Reference Documentation
  - https://www.atlassian.com/git/tutorial - Git Tutorial
  - http://gitready.com/ - Git Tutorial

- Workflows
  - https://www.atlassian.com/git/workflows - Tutorial on common Git workflows
  - http://yakiloo.com/getting-started-git-flow/ - About Git Flow (advanced topic)
- Getting Help
  - http://stackoverflow.com/ - All things programming
  - https://help.github.com/ - Git Recipies

Cambridge Technology Partners

First Repository

Creating a Repository

```
$ mkdir myrepo
$ cd myrepo
$ git init
$ git ls -la
```

git init

Do this in one swoop with

```
$ git init myrepo
```

Adding Files

```
$ touch index.html
$ git status
$ git add index.html
$ git status
```

git status

git add

`git add` works also with patterns:

```
$ git add ‘*.java’
$ git add .
$ git add folder/
```

You can even stage parts of a file

```
$ git add –p|-i
```

Stage all changes (including deleted files) in the working directory with

```
$ git add -A .
```

Committing Files

```
$ git commit
$ git status
$ git log --oneline --decorate
```

git commit

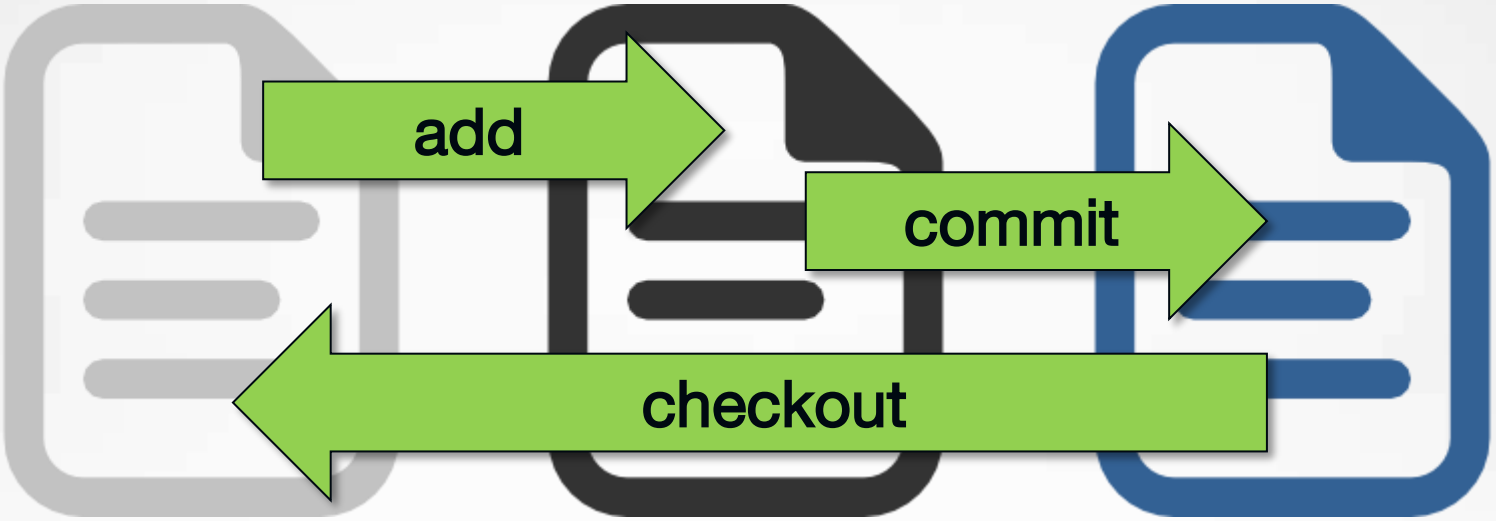
Commit directly with commit message:

```
$ git commit -m ‘Been there, done that’
$ git commit -am ‘Add also modified files directly’
```

Need a different commit editor?

```
export EDITOR=vim
```

Three Stage Thinking



Working Directory

Staging Area

Repository History

Deleting and ignoring Files

```
$ touch test1.log test2.log
$ git add test1.log
$ git commit
$ vim .gitignore
$ git status
$ git rm test1.log
$ git commit
```

.gitignore

git rm

A shell script for easily accessing - **gi**tignore **bo**ilerplates
https://github.com/simonwhitaker/gitignore-boilerplates

```
$ gibo Java Eclipse >> .gitignore
```

How does my repo look like?

Cambridge Technology Partners

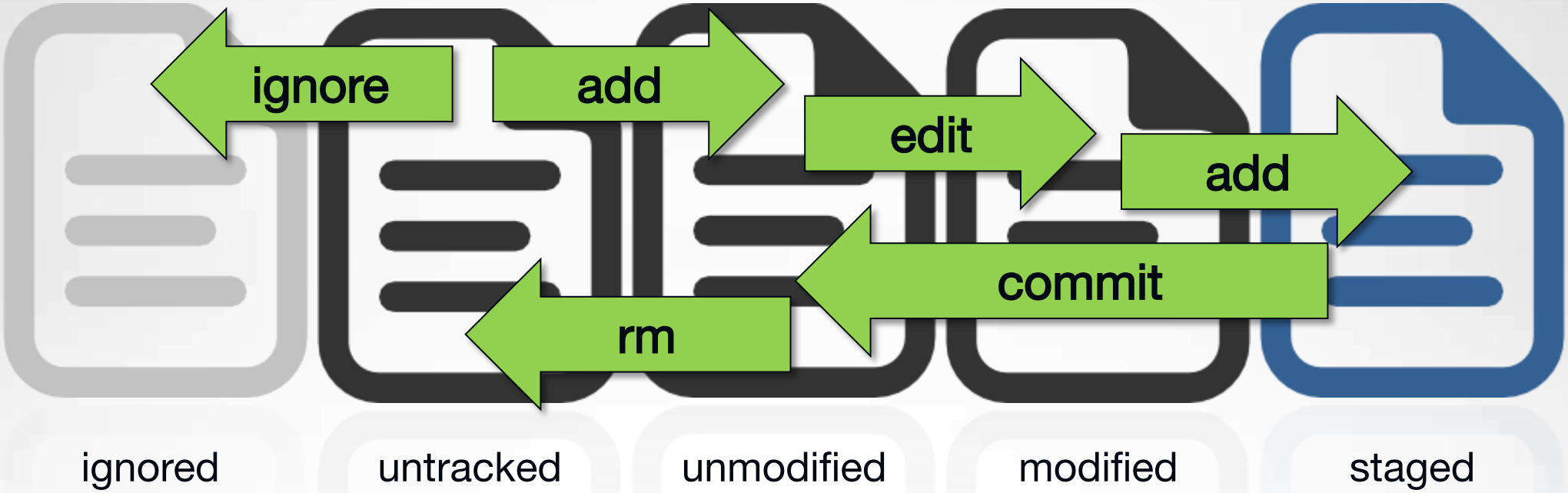
`git log` gives you and overview of your repository structure.

```
$ git log
$ git log -p
$ git log --oneline --decorate

$ git log --graph --pretty=format:'%Cred%h%Creset
-%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold
blue)<%an>%Creset' --abbrev-commit
```

The Git File Workflow

Cambridge Technology Partners



Cambridge Technology Partners

Branching and Merging

Branching

```
$ git branch mybranch
$ git branch
$ git checkout mybranch
```

git branch

Delete the branch with

```
$ git branch -d mybranch
$ git branch -D mybranch      # if unmerged
```

Create a branch and check it out in one swoop

```
$ git checkout -b mybranch
```

Time for some serious work

Objectives: Getting familiar with branching and tagging.

- Create new branch and modify repository
- Switch between branches
- Delete branch
- Tag commits

```
$ git branch
$ git checkout
$ git tag
```

I'm not done yet

```
$ git status    # staged stuff
$ git stash
$ git status
…
$ git stash list
$ git stash apply [--index]
$ git stash drop stash@{0}
```

git stash

Apply and remove stash in one swoop

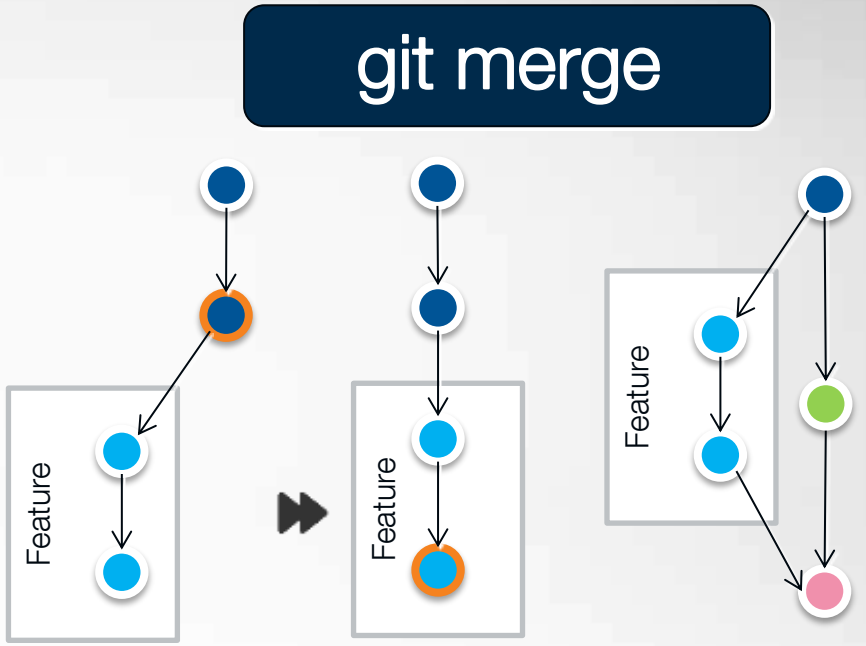
```
$ git stash pop
```

Merging

```
$ git checkout master
$ git branch mybranch
$ git showlog
$ git branch
```

Fast-forward is default

```
$ git merge --no-ff
```



Deactivating fast-forward merges per branch

```
$ git config branch.master.mergeoptions  "--no-ff"
```

Diffs

Cambridge Technology Partners

```
$ git diff mybranch master
```

git diff

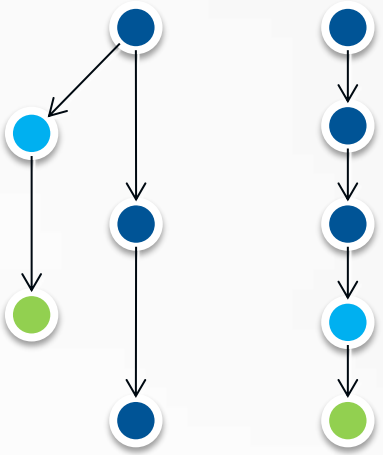
Diff works also on the branch history

```
$ git diff                    # unstaged
$ git diff HEAD^^ HEAD        # from to
$ git diff hash1...hash2      # from to
```

Rebasing

```
$ git checkout master
$ git rebase mybranch
```

git rebase



Rewriting history: Interactive rebase last four commits

```
$ git rebase --i HEAD~4
```

All your base are belong to us

Objectives: Learn how rebase works.

- Make changes on selected branch and rebase it with master

```
$ git rebase <BRANCH>
```

Cambridge Technology Partners

Going Remote

Cloning Repositories

```
$ git clone [#remote]
```

git clone

Clone into a specific or existing (empty) folder

```
$ git clone [#remote] myclonedrepo
```

Git Protocols

Cambridge Technology Partners

- ssh / git: Securely connect to remote machines

```
git clone git@github.com:ctpconsulting/jazoon-git-workshops.git
```

- HTTPS: Firewall friendly

```
git clone https://github.com/ctpconsulting/jazoon-git-workshops.git
```

- File – simple. Can be used with e.g. a shared drive

```
git clone file:///home/thug/repo/chopen-workshop-git
```

Cloning directly without the file protocol will use hard links

```
$ git clone /home/thug/repo/jazoon-git-workshops
```

Remotes

Cambridge Technology Partners

```
$ git init myremoterepo
$ cd myremoterepo
$ … # commit something
$ git remote add origin [#remote]
$ git remote -v
```

git remote

Git is distributed – you can have more than one remote!

```
$ git remote add https-origin https://myrepo.com/repo.git
```

Submitting Changes

```
$ git push -u origin master
$ …
$ git push
```

git push

Forced push

```
$ git push --force
```

By default, Git always tries to push all matching branches.
Configuration to push only current to upstream:

```
$ git config push.default upstream
```

Retrieving Changes

```
$ git fetch
$ git merge origin/master
```

Or short-hand

```
$ git pull
```

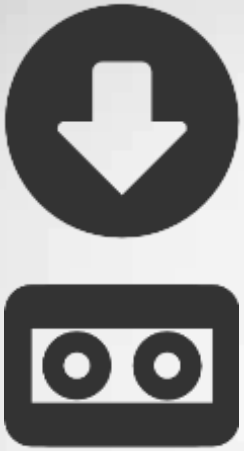
git fetch

git pull

Resolution strategy for merge conflicts

```
$ git pull -Xours
$ git pull -Xtheirs
```

Retrieving Changes



```
$ git pull --rebase
```

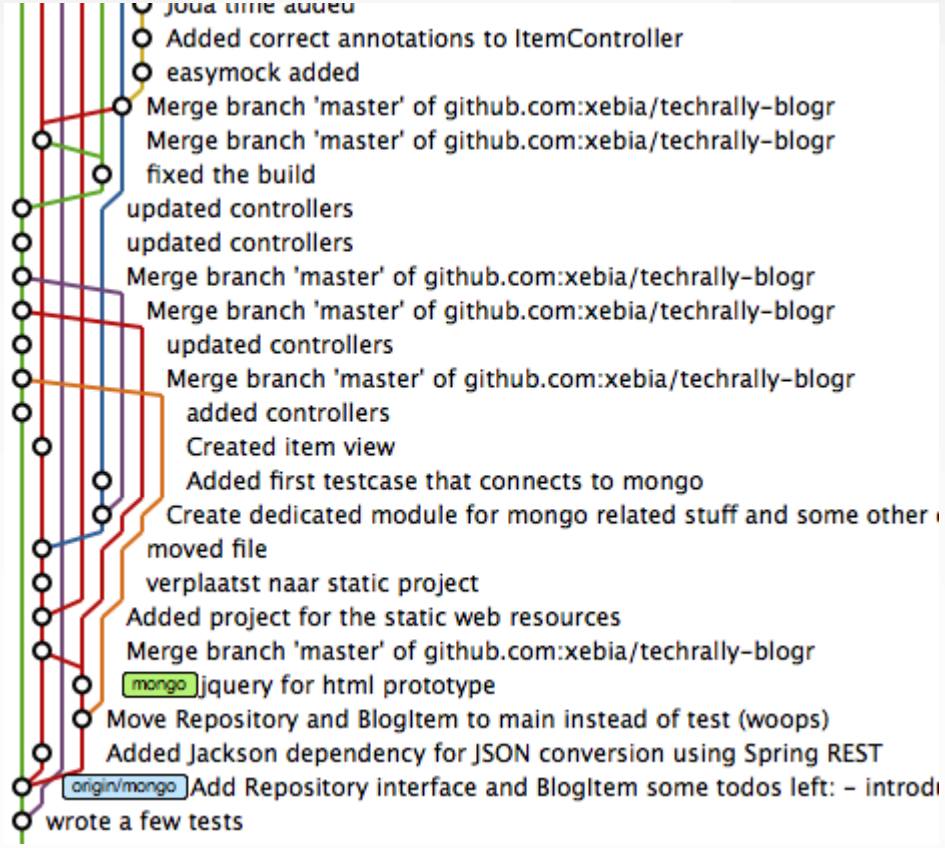
During a regular daily workflow where several team members sync a single branch often, the timeline gets polluted with unnecessary micro-merges on a regular `git pull`. Rebasing ensures that the commits are always re-applied so that the history stays linear.

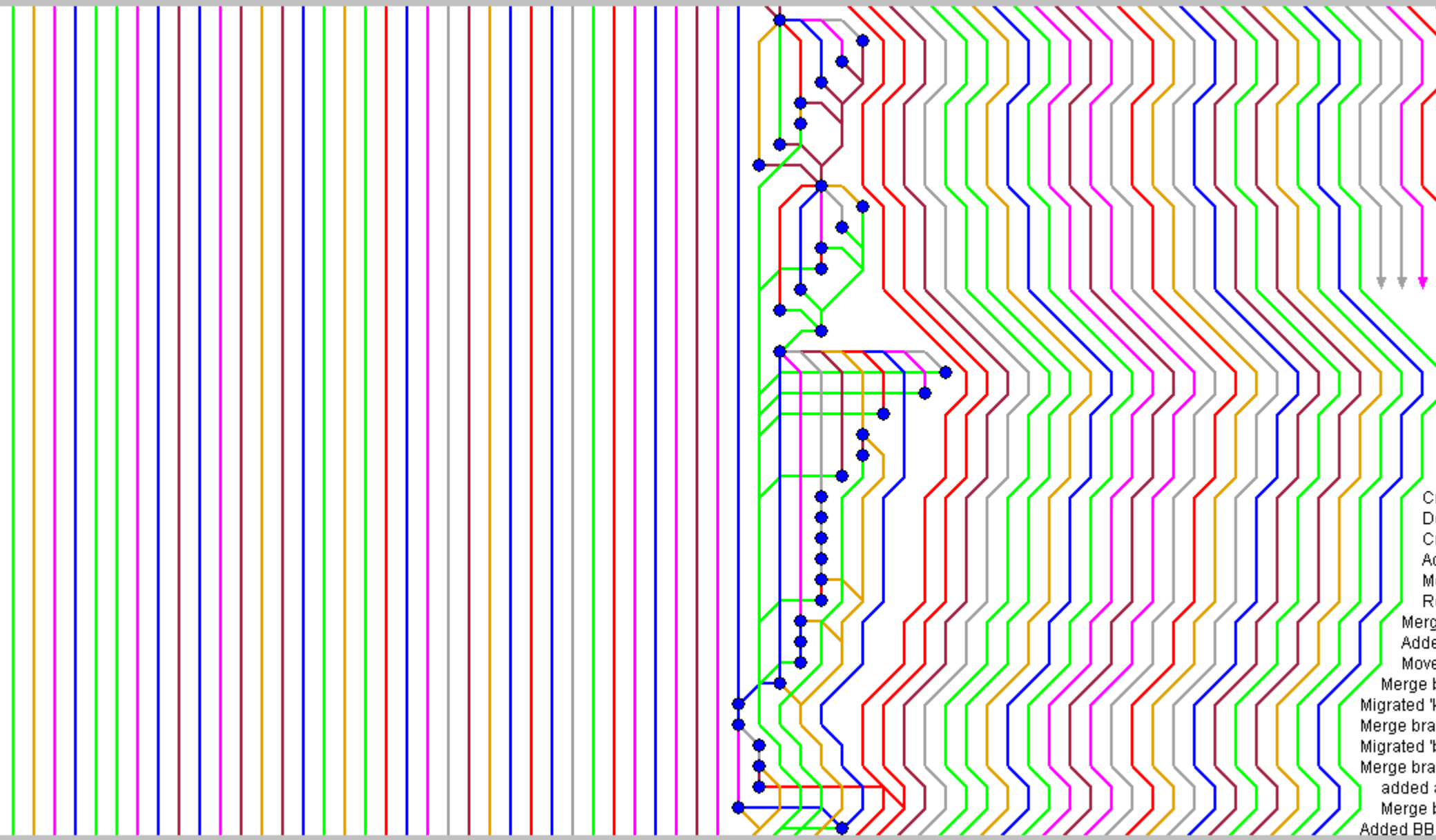
Make `git pull` on master always use rebase

```
$ git config branch.master.rebase true
```

Or make it a default for every tracking branch strategy

```
$ git config --global branch.autosetuprebase always
```




Merg
Adde
Move
Merge b
Migrated '
Merge bra
Migrated '
Merge bra
added
Merge b
Added BB

Get IT right

Thank you!

Credits

- Icons provided by Icons8: http://icons8.com/