Oh my GIT!

Git ate my work and other stories

Vojtěch Vladyka

December 10, 2019

Table of contents

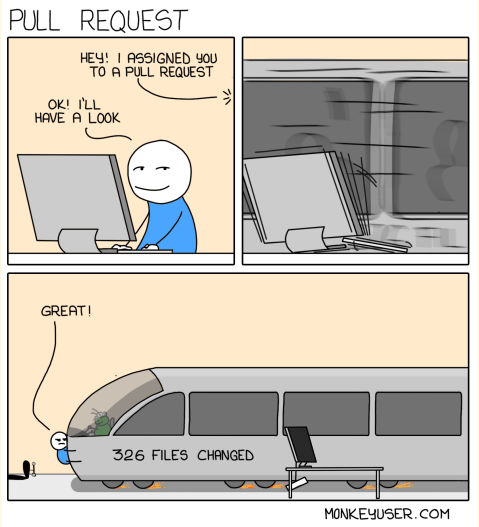
What is git?

- Git is distributed by design
- Works with whole sourcetree (unlike SVN which works with individual files and their revisions) and always capture state of working tree
- Strong support for non-linear development → rapid branching & merging
- Cryptographic authentication of history - every commit has SHA-1 hash of everything leading to that point
- Version control system made by Linus Torvalds (mostly) at 2005 for Linux Kernel development
- Supports rapid branching & merging
- it is pronounced GIT with G like GIF[1]

[1] Google. *Tech Talk: Linus Torvalds on git*. URL: https://www.youtube.com/watch?v=4XpnKHJAok8.

Git philosophy

- Create branch for each feature
- Commit often
- Merge back to master as soon as possible
- Create small isolated as-much-as-possible pull requests



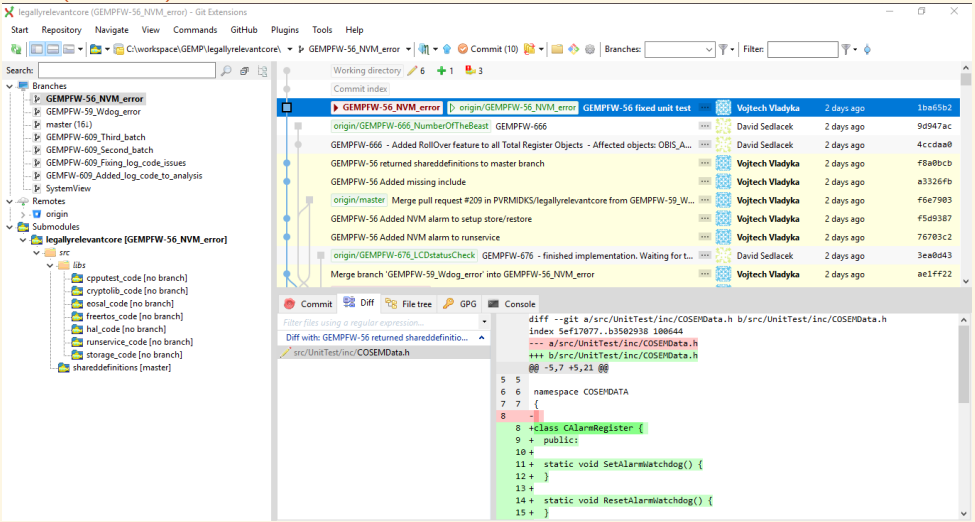
Setup git

Git clients

1. Git (all platforms)
2. Git Extensions (Windows, Mac)
3. Tortoise Git (Windows)
4. Sourcetree (Windows)
5. Fork (Windows, Mac)
6. Gitkraken (all platforms)
7. ...many others

Setup git

Git Extensions (Git client)



Download it here → `https://gitextensions.github.io`

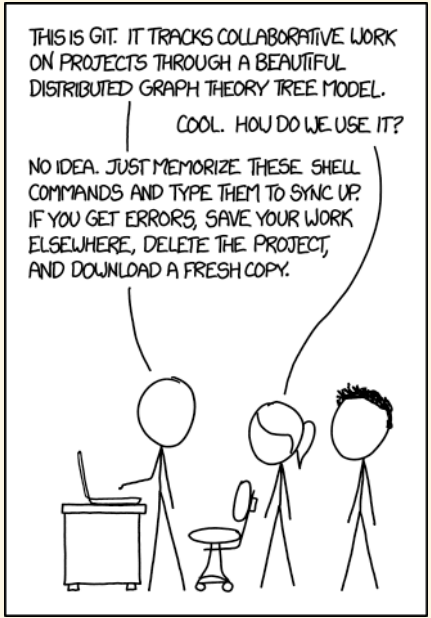
Git essentials

Short and incomplete command overview

```
git init
    clone

    fetch
    merge
    push
    pull

    branch
    checkout
    merge
    branch -d
```

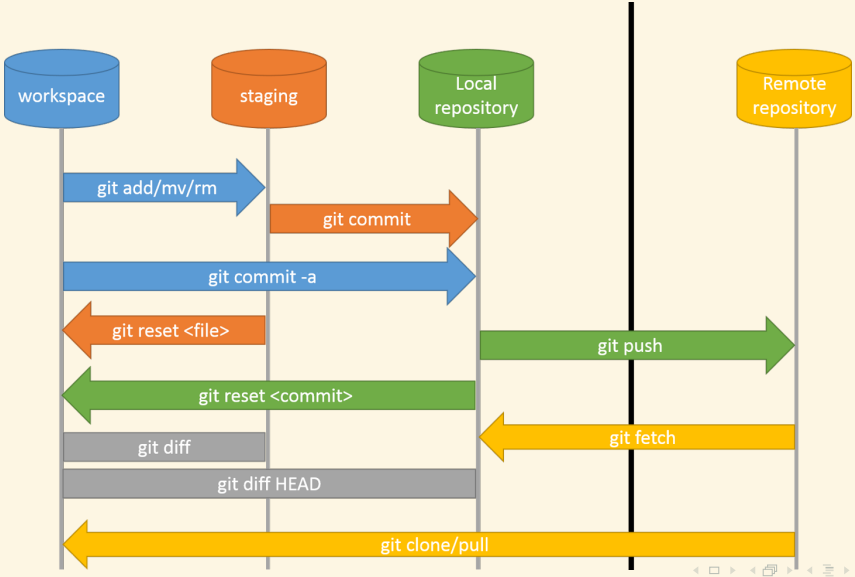
```
git log
    log --follow [file]
    reflog

    add
    commit

    reset
    reset --hard
    stash

    rebase
    cherrypick
```

Git essentials



Git essentials

Workflow overview



Git essentials

Init repo

```
git init
```

→ initialize new repository in current directory with all its contents

```
git clone https://somerepo.com/repo.git
```

→ downloads repository with all current branches & commits

Git essentials

Fetching changes and returning them back

```
git pull
```

→ Download commits from server and apply them to your working tree

```
git push
```

→ Upload your commits to server

```
git fetch --all
```

→ Download commits from server (from all branches eventually)

```
git commit [files] -m "Commit message"
```

→ Create a commit with commit message and specified files
→ You can use -a to commit all changes instead list of files

```
git add [file]
```

→ Move file to staging area - add it to commit list

Git essentials

Git essentials

Undoing changes

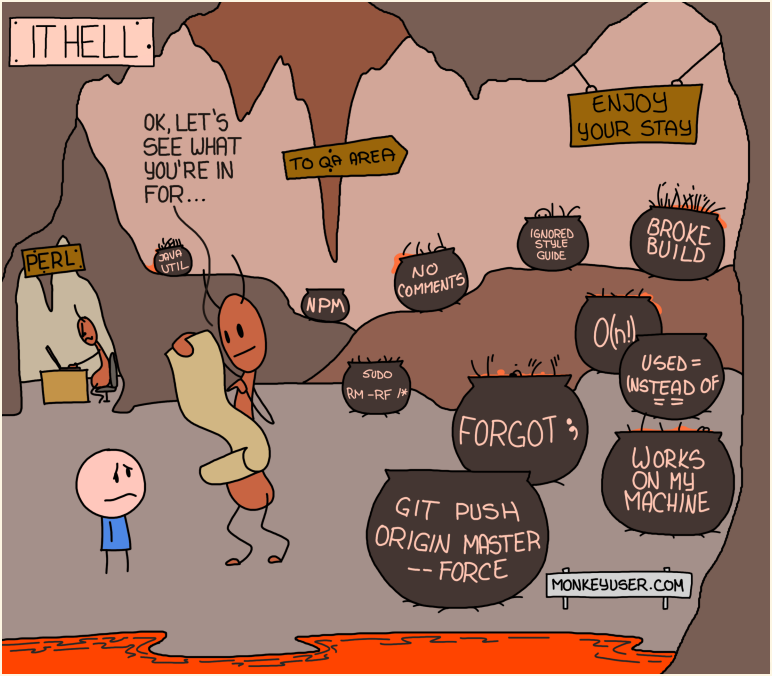
```
git reset
```

→ Soft reset - remove files from staging area and move them back to working area. Nothing is actually reverted, this resets it only for git.

```
git reset --hard
```

→ Hard reset - this do all what soft reset but also REMOVES your uncommitted changes.

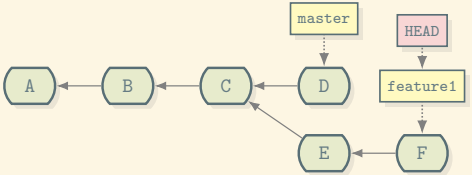
→ You can do this only locally. Once you push your changes, you cannot get it back. (not exactly true but you will break a lot of thing for all colleagues). Only way how to propperly revert things it using git revert.


IT HELL
OK, LET'S SEE WHAT YOU'RE IN FOR...
TO QA AREA
ENJOY YOUR STAY
PERL
JAVA UTIL
NPM
NO COMMENTS
IGNORED STYLE GUIDE
BROKE BUILD
O(n!)
SUDO RM -RF /*
USED = INSTEAD OF ==
FORGOT ;
WORKS ON MY MACHINE
GIT PUSH ORIGIN MASTER --FORCE
MONKEYUSER.COM

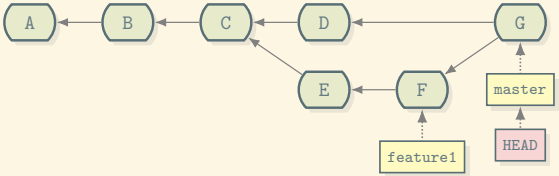
Git essentials

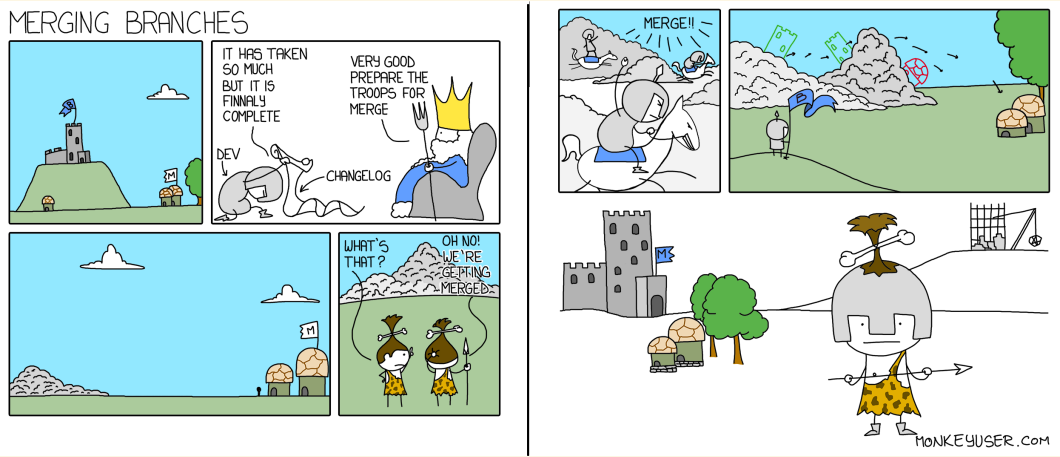
Branching

```
git checkout -b feature1
(git branch feature1 ; git checkout feature1)
```




```
git checkout master
git merge feature1
```




MERGING BRANCHES
IT HAS TAKEN SO MUCH BUT IT IS FINNALY COMPLETE
VERY GOOD PREPARE THE TROOPS FOR MERGE
DEV
CHANGELOG
WHAT'S THAT?
OH NO! WE'RE GETTING MERGED
MERGE!!
MONKEYUSER.COM

Git essentials

Logging and searching lost stuff

```
git log --graph
```

→ Shows log for whole repository tree

```
git log --follow [file]
```

→ Shows log for selected file even across renames

```
git reflog
```

→ Shows log of ALL changes in repository. Stashes, detached branches, all...

Git essentials

Stashing - let me just put this here for later (and never use it again)

```
git stash
git stash push
```

→ Stashes current changes to new stash stack

```
git stash pop
```

→ Pops newest stash and apply it to current working copy

```
git stash list
```

→ Show all stashes in list

```
git stash apply stash@{2}
```

→ Get stash #2 and apply it to current working copy

```
git checkout stash@{2} -- somefile
```

→ Get stash #2 and apply it to specified files in current working copy

Git essentials

Submodules

```
git submodule add <url to submodule repository>
```

→ This adds new submodule to current repository and also clones it.

```
git submodule init
```

→ This clones submodules when you just cloned new repository with submodules.

```
git submodule update --init --recursive
```

→ And this downloads submodules to version specified in repository (and eventually clones new submodules)

```
git submodule foreach 'do something'
```

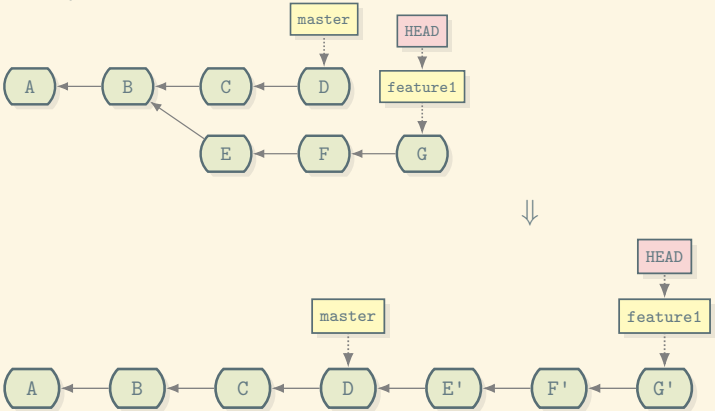
→ This runs 'do something' on every submodule.

Git essentials

Rebase

```
git rebase target_branch source_branch
```

→ Take source branch and reattach it on end of target branch. This creates series of new commits and remove old ones from tree (althrough they will still be accessible by reflog)

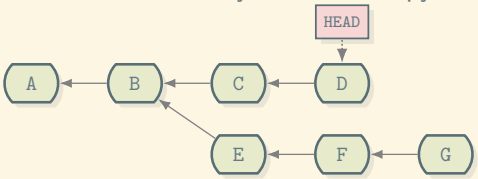


Git essentials

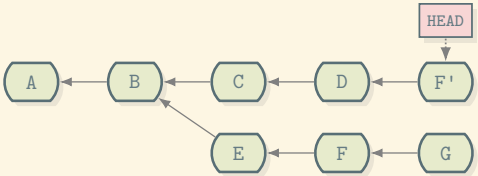
Cherry picking

```
git cherry-pick <commit hash>
```

→ Select commit by hash and copy it on end of current branch as new commit.



⇓



Scenarios

It's time to break some stuff.

Scenarios

Fresh start - cloning

```
git clone https://bitbucket.honeywell.com/users/e878714/
          repos/playground/commits
```

→ This downloads whole repository into current working directory. (that URL should not be splitted but slide is too short...)

Scenarios

Branching

→ We do not want to fight all together so we create own branches

```
git branch your-branch-name
git checkout your-branch-name
```

→ Now we should all be at 'your-branch-name' branch.

Scenarios

Committing

→ Do some changes (anything) in code you just cloned

```
git commit -am "Your commit message"
```

→ This commits all your changes with your commit message.

Scenarios

Pushing

→ Let's put our changes on server

```
git push
```

→ This should actually fail because we are pushing to non-existen branch on server

```
git push --set-upstream origin your-branch-name
```

→ And this will work. It creates new branch with our name on remote server named 'origin'

Scenarios

Adding

→ Let's create new file and add it to repository

```
git add our-new-file.txt
```

→ This adds our-new-file.txt to staging area. We need to commit it next.

```
git commit -am "Your commit message"
```

→ If we are happy with it, we can push it.

```
git push
```

Scenarios

Merging and pulling

→ Now we want to merge our stuff into master. Normally we will create pull request but today we will merge direcly to master. We start with checking out master branch.

```
git checkout master
```

→ Now we need to get latest changes from it.

```
git pull
```

→ And now we can merge it.

```
git merge vojtuv-test
```

→ And push result.

```
git push
```

Scenarios

Log, Checkout

→ We want to work on our branch again (or somebody elses branch).
→ We get latest changes firts from all branches and delete references to non-existent ones.

```
git fetch --all --prune
```

→ Now we can take a look on current repository log

```
git log --graph
```

→ And eventually checkouts something

```
git checkout our-branch-name
```

Scenarios

Stash

→ Let's do some changes...
→ Ok, we don't want them actaully. We need to get rid of them. But it's really nice piece of work so we will keep them for later.

```
git stash
```

→ Ok, it's gone. You can verify it with status.

```
git status
```

→ Now fast forward a little bit and assume we want them back. We can be on different branch, we can be in future, it doesn't matter.

```
git stash list
git stash apply stash@{0}
```

→ Check who's back!

Scenarios

Reset

→ Oh my god, that change was rubbish... We need to delete it.

```
git reset --hard
```

→ Uff, it's gone.

Scenarios

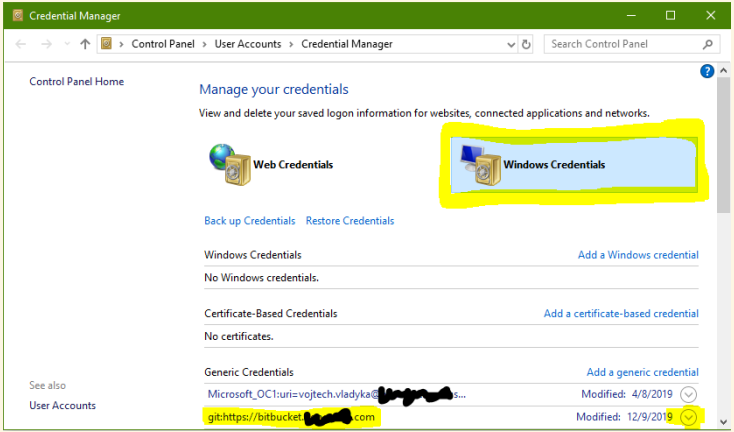
Git online playground

If you want to play with branching, rebasing, merging, squashing etc., this is great tool for playing with it.
`https://learngitbranching.js.org`

Troubleshooting

Windows Crenedital Manager

Git.exe uses Credential manager for storing login information. It's nice feature with one undesired issue - password change doesn't affect stored credentials so you need to remove record manually when you change your password.



Resources

- Google. *Tech Talk: Linus Torvalds on git*. URL: https://www.youtube.com/watch?v=4XpnKHJAok8.
- Mark Erikson. *Git Under the Hood*. URL: https://blog.isquaredsoftware.com/presentations/2019-03-git-internals-rewrite/#/0.
- GitHub. *Git Cheat Sheet*. URL: https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf.
- Git community. *Git –distributed-even-if-your-workflow-isnt*. URL: https://git-scm.com.

Q & A

Thank you for your attention.