



3 – GIT, the Version Control System

Robotics and Computer Vision (BPC-PRP)

Course supervisor:

Ing. Adam Ligocki, Ph.D.

Slides:

Ing. Adam Ligocki, Ph.D.



Robotics and AI

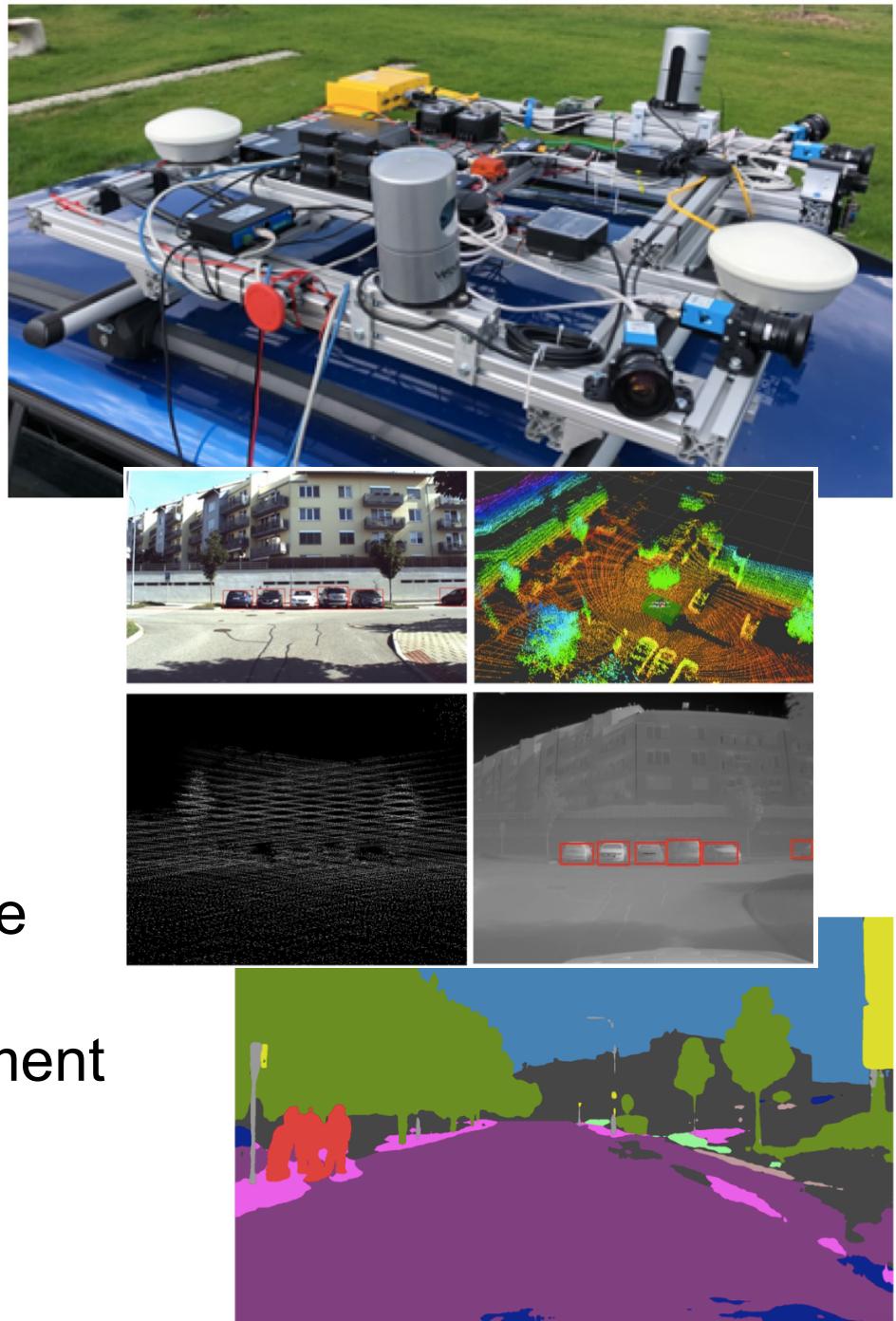
Ing. Adam Ligocki, Ph.D.

Position: Research Staff

Research: Data Fusion

Room: SE1.102

Profile



Web: <https://www.vut.cz/lide/adam-ligocki-154791>



Git – Distributed System for Version Control

System for versioning the source codes (generally any text sources).

Allows to track changes in the project during the time (who and when made which change).

Keeps backup of the project.

Allows multiple developers collaboration.



Snapshot - In Git, a **snapshot** is the full state of your repository at a specific moment. Each **commit** captures this state, even though Git optimizes storage by reusing unchanged files.

Commit is a Git object that encapsulates a **snapshot** of your repository along with important metadata. Each commit contains:

- **Reference to the tree**
- **Reference to the parent**
- **Metadata**
- **Hash**



Mr Pokee / Instagram



Repository (repo) is a collection of files and their complete history managed by Git. This includes all the commits, branches, and tags that represent changes over time.

A **Repository** can exist on your local computer (local repo) or on a remote server (remote repo). While "origin" is often used as the default name for a remote repository when you clone a repo, it's simply a convention.

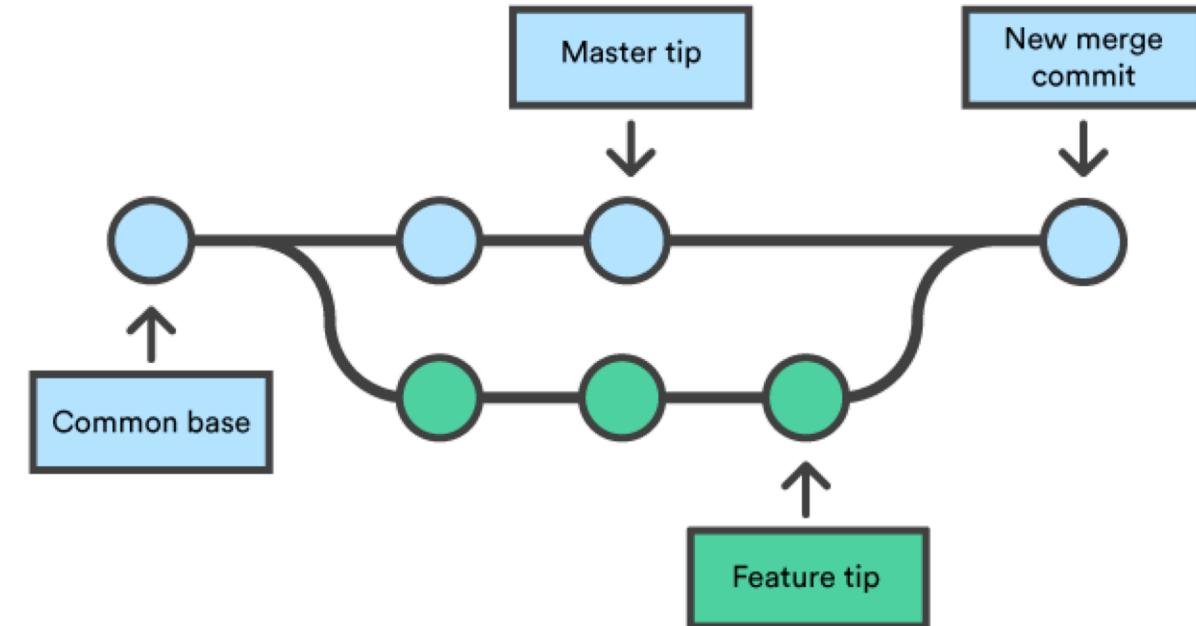
Cloning is the process of copying an entire repository, including its history, from a remote location to your local machine.

Pulling downloads commits from the remote repository and integrates them into your current branch. Technically, a pull is a combination of a fetch (downloading the commits) followed by a merge (integrating the commits).

Pushing is the act of uploading your local commits to a remote repository, updating it with your changes.

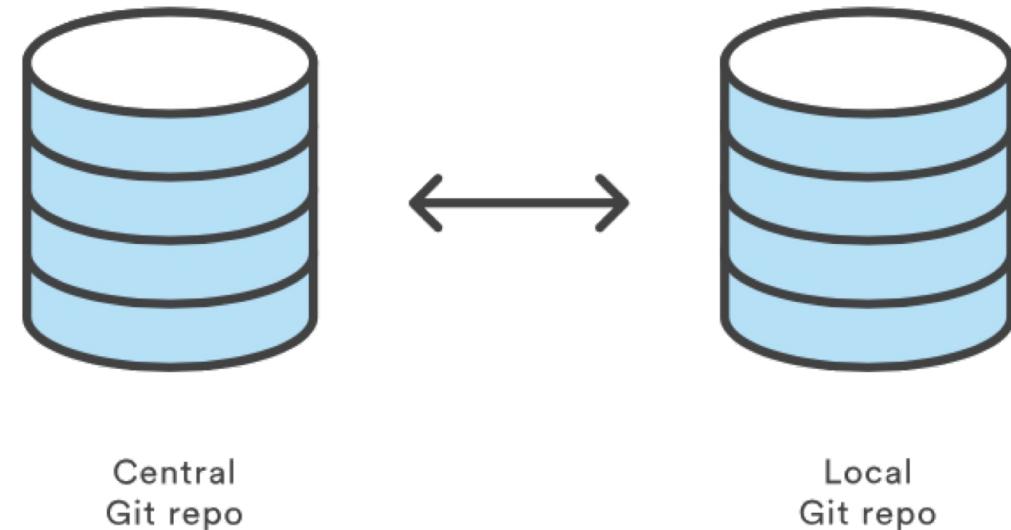
All **commits** in Git are organized into **branches**. A repository starts with a default branch (usually called **master** or **main**).

Branching lets you work on features independently, and merging brings those changes together into one branch.



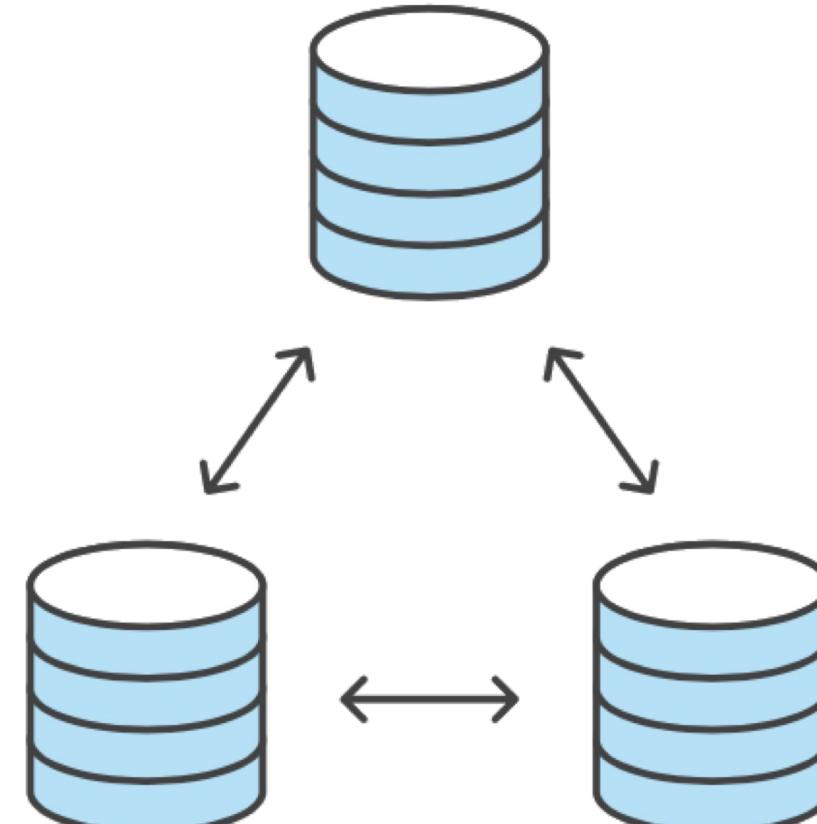
You write your code and, after completing each small task, you save your changes (**diff**) with a **commit** in your local repository.

Once a larger feature is complete, you push your commits to a remote repository (commonly called **origin**), where other developers can pull and integrate them.



Git is fully decentralized, meaning every repository can be a complete copy of the entire project, including all branches and history.

This redundancy ensures that if a server fails, any repository can restore the complete history.



<https://www.atlassian.com/git/tutorials>



.gitignore File

File that defines files or folders that GIT should ignore

Typically the caches, build dependent files or binaries

Never commit binaries into GIT!

For .gitignore templates see:

<https://github.com/github/gitignore>

or

<http://gitignore.io>

```
git_ignore_project ~/Developer/git_ignore_project
main
Project
git_ignore_project ~/Developer/git_ignore_project
  cmake-build-debug
    .gitignore
    CMakeLists.txt
    main.cpp
...
External Libraries
Scratches and Consoles
Terminal Local + 
adashligocki@Adashs-Air git_ignore_project % git init
Initialized empty Git repository in /Users/adashligocki/Developer/git_ignore_project/.git/
adashligocki@Adashs-Air git_ignore_project % git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
    CMakeLists.txt
    main.cpp

nothing added to commit but untracked files present (use "git add" to track)
adashligocki@Adashs-Air git_ignore_project % ls -la
total 24
drwxr-xr-x  8 adashligocki  staff  256 Feb 16 11:14 .
drwxr-xr-x 18 adashligocki  staff  576 Feb 16 11:12 ..
drwxr-xr-x  9 adashligocki  staff  288 Feb 16 11:14 .git
-rw-r--r--  1 adashligocki  staff   35 Feb 16 11:14 .gitignore
drwxr-xr-x  7 adashligocki  staff  224 Feb 16 11:13 .idea
-rw-r--r--  1 adashligocki  staff  138 Feb 16 11:13 CMakeLists.txt
drwxr-xr-x 11 adashligocki  staff  352 Feb 16 11:13 cmake-build-debug
-rw-r--r--  1 adashligocki  staff   97 Feb 16 11:13 main.cpp
adashligocki@Adashs-Air git_ignore_project %

git_ignore_project > .gitignore
```

Git works well for the text file version control. It tracks newly added, deleted, or modified lines of text (source code)

Git is not able to effectively follow changes in the binary files (exec, libraries, compressed files, etc ...) or the automatically generated files (the build/ folder)

Use **.gitignore** file in the root of the repository to exclude redundant files.

For large files in git, see

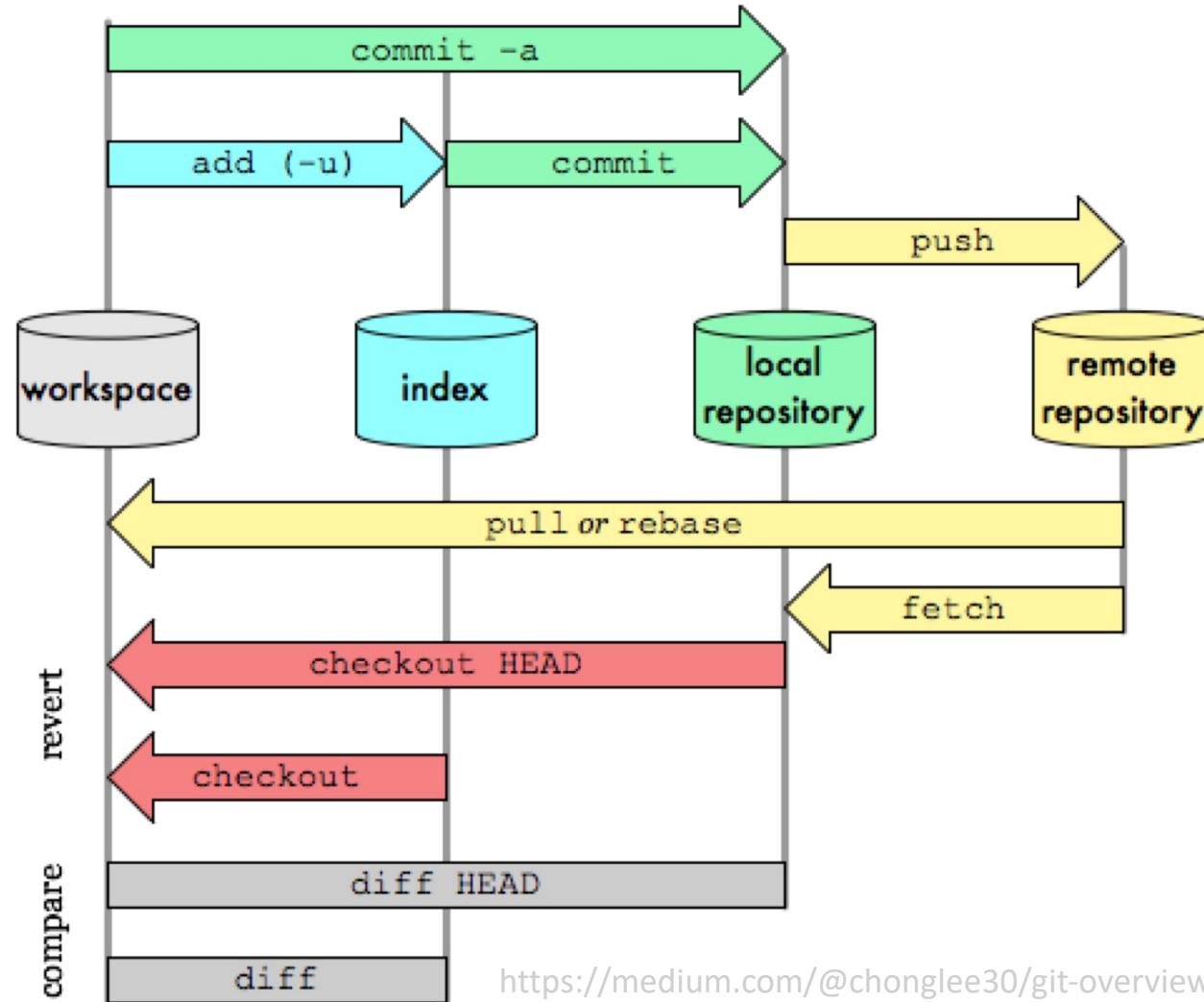
Large File Storage (LFS)



Minecraft Wiki

Git Data Transport Commands

<http://osteale.com>



Git is an open-source project, and it is provided as a precompiled binary for all three major OS platforms

On Debian Linux dist. use:

```
sudo apt install git
```

For details see the official web:

<https://git-scm.com/downloads>

The screenshot shows the official Git website at <https://git-scm.com/>. The page features a large header with the Git logo and the tagline "local-branching-on-the-cheap". A search bar is located in the top right corner. The main navigation menu includes links for "About", "Documentation", "Downloads" (which is highlighted in red), and "Community". The "Downloads" section is expanded, showing links for "Mac OS X", "Windows", and "Linux/Unix". To the right of this section is a large image of a Mac desktop monitor displaying the latest source release version 2.25.1 and a "Download 2.23.0 for Mac" button. Below the download links, there is a note about older releases and the Git source repository on GitHub. The page also includes sections for "GUI Clients", "Logos", and "Git via Git", each with its own descriptive text and links.

git init - initialize local repository in the local directory, where your project is located

git add - this command adds files to the index
(now the changes are accepted and waiting to be committed)

git commit - saves the staging changes to the local repository
each commit has a unique SHA-1 20-byte hash that works as an alias

git clone - downloads an identical copy of the remote repository to your computer

git push - sends all the new commits from your local repository to the remote repository

git pull - downloads new commits from remote repository to the local repository

git remote - command for configuring connection with a remote repository



<https://unsplash.com/s/photos/cute-cat>

git diff - shows all the changes in the working directory that has not been added to the index yet

git status - printout the status of the repository

git log - simple visualization of the repository history

git stash - saves your current changes and cleans working directory

git branch - allows you to create separated branches, where you can develop independent features

git merge - combines some branch with the current one

git checkout - switch your repository to the state that corresponds with the given commit (hash)
checkout allows you also to switch between branches



<https://www.pexels.com/search/cute%20puppy/>



GIT - Conflicts

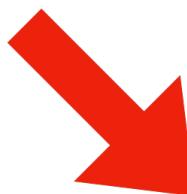
```
#include <stdio.h>
int main() {
    doEvenCoolerStuff();
    return 0;
}
```



```
#include <stdio.h>
int main() {
    return 0;
}
```



```
#include <stdio.h>
int main() {
    doSuperCoolStuff();
    return 0;
}
```

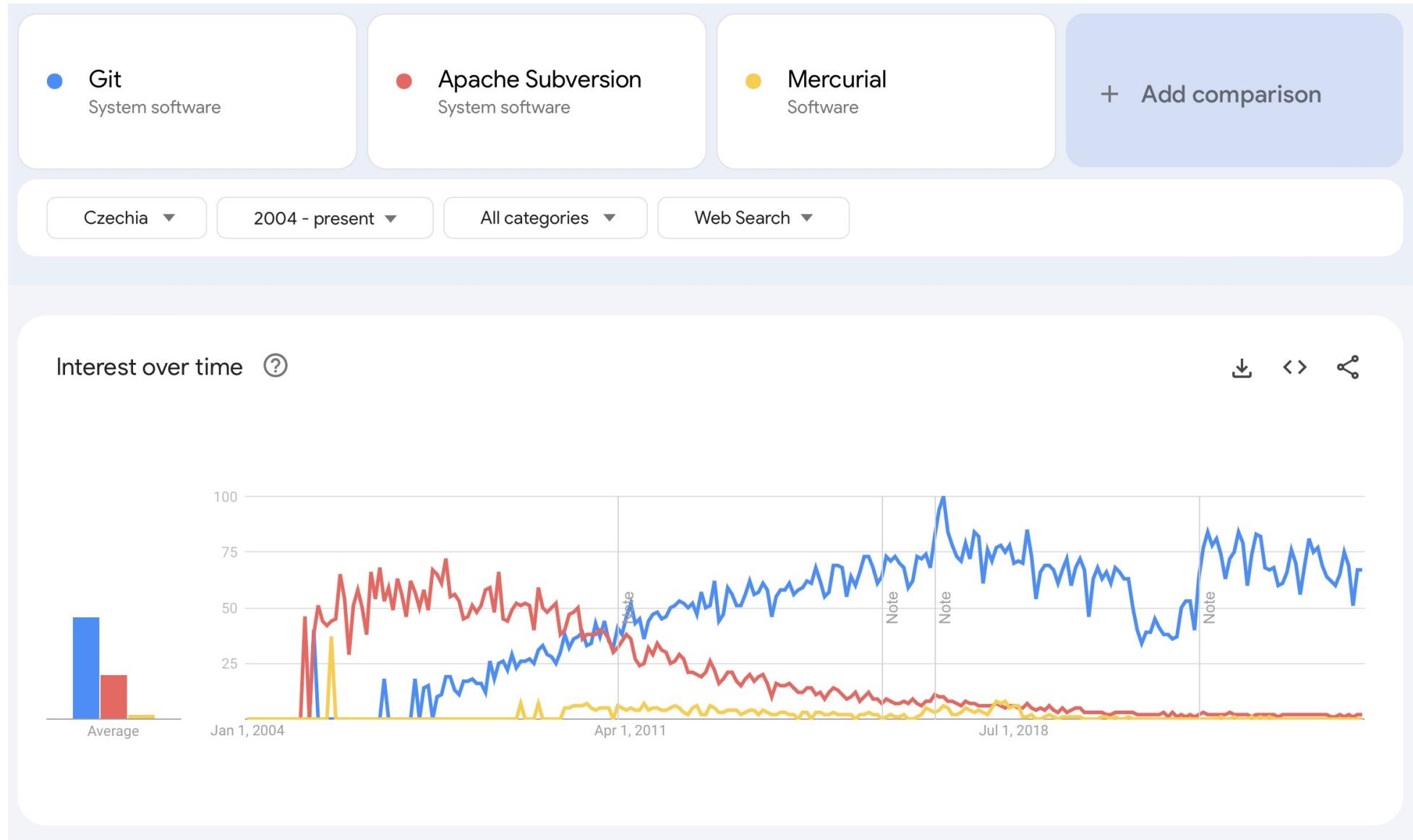


```
#include <stdio.h>
int main() {
    <<<<< HEAD
        doSuperCoolStuff();
    =====
        doEvenCoolerStuff();
    >>>>>
        return 0;
}
```





Why does GIT matter?



There are several online services that provides git functionality.

Additionally, these services allow you to configure the repository, track issues, create wiki pages, perform continuous integration tests, etc.

For non-commercial usage, it is free

Can deploy GitLab on your own server





GIT – VCS integration in CLion

The screenshot shows the CLion IDE interface with the following details:

- Top Bar:** Shows project names "git_ignore_project" and "main", build configurations "Debug", and various tool icons.
- Left Sidebar:** A tree view titled "Commit" showing "Changes 3 files": ".gitignore", "CMakeLists.txt", and "main.cpp".
 - "Amend" button is available.
 - "Initial commit" message is present.
 - "Commit" and "Commit and Push..." buttons are at the bottom.
- Central Area:** Code editor tabs for "CMakeLists.txt", "main.cpp", and ".gitignore". The "main.cpp" tab is active, displaying the following code:

```
#include <iostream>
int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```
- Bottom Left:** Git integration sidebar with buttons for "HEAD (Current Branch)", "Local", and "main". It also includes navigation icons for file operations like Open, Save, Find, and More.
- Bottom Center:** Git log table with columns for "Q", ".* Cc", "Branch", "User", "Date", "Paths", and "Actions". It displays "No changes committed." and a "Commit local changes (⌘K)" button.
- Bottom Right:** "Commit details" section.
- Bottom Status Bar:** Shows "clang-tidy" 7:1, LF, UTF-8, 4 spaces, C++: git_ignore_project | Debug, and a file icon.



GIT – VCS integration in CLion

The screenshot illustrates the Git integration within the CLion IDE. The interface includes:

- Top Bar:** CLion, File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, Git, Window, Help.
- Project View:** Shows a folder named "Commit" and a file named ".gitignore". The ".gitignore" file contains the text "World!" << std::endl;".
- Central Area:** A large text editor window titled "Commit" with the message "Initial commit".
- Git Operations Pop-up:** A context menu for the commit operation, listing options like Commit..., Push..., Update Project..., Pull..., Merge..., Rebase..., Branches..., New Branch..., New Tag..., Reset HEAD..., Show Git Log, Patch, Uncommitted Changes, Current File, GitHub, Manage Remotes..., and Clone....
- Bottom Panel:** A Git log window showing a single commit: "Initial commit" by "adam ligocki" at "A minute ago". It also displays a message to "Select commit to view changes" and "Commit details".
- Bottom Status Bar:** .clang-tidy 7:1 LF UTF-8 4 spaces C++: git_ignore_project | Debug



Git Cheat Sheet

For more awesome cheat sheets
visit [rebellabs.org!](http://rebellabs.org/)



Create a Repository

From scratch -- Create a new local repository

```
$ git init [project name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit]:[file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working with Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b
```

```
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

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

Push local changes to the origin

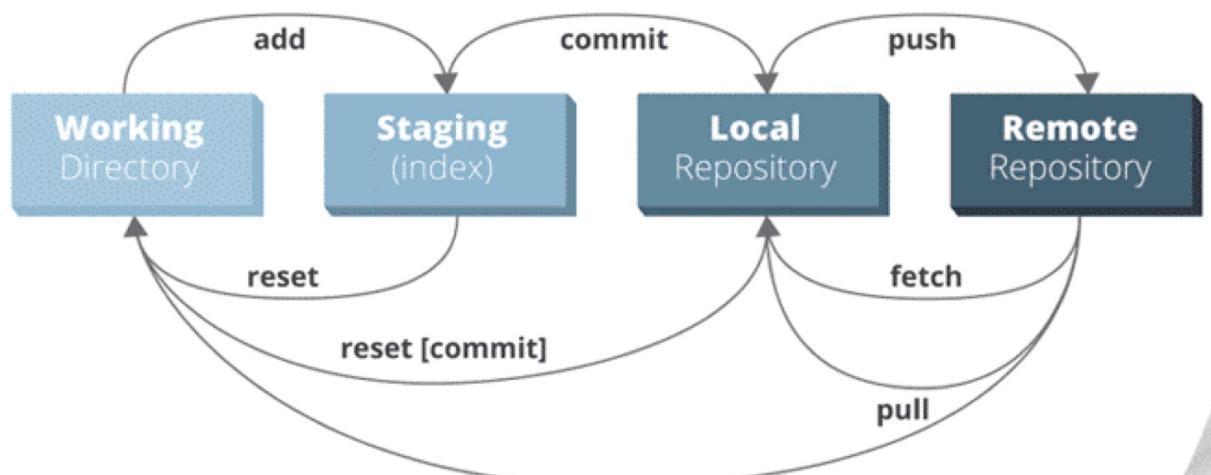
```
$ git push
```

Finally!

When in doubt, use git help

```
$ git command --help
```

Or visit <https://training.github.com/> for official GitHub training.





git init

git add

git commit

git status

git checkout

git push

git pull

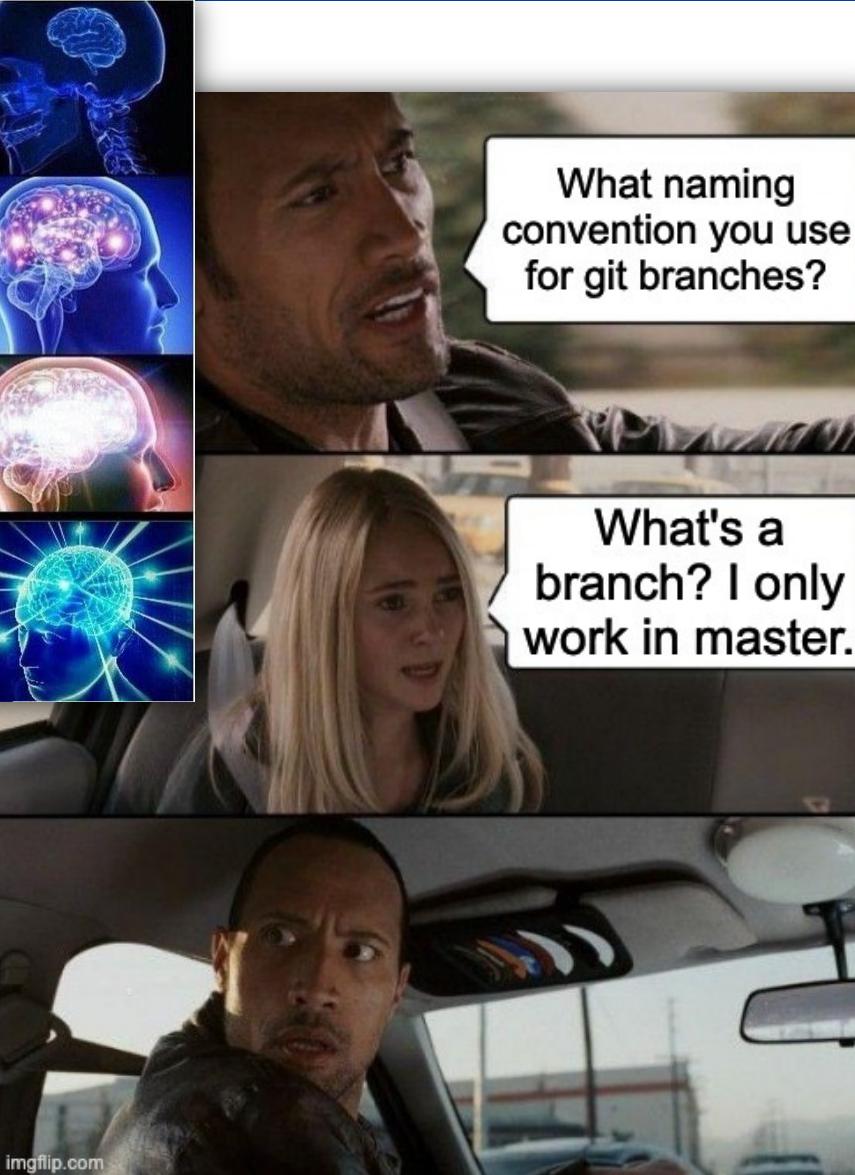
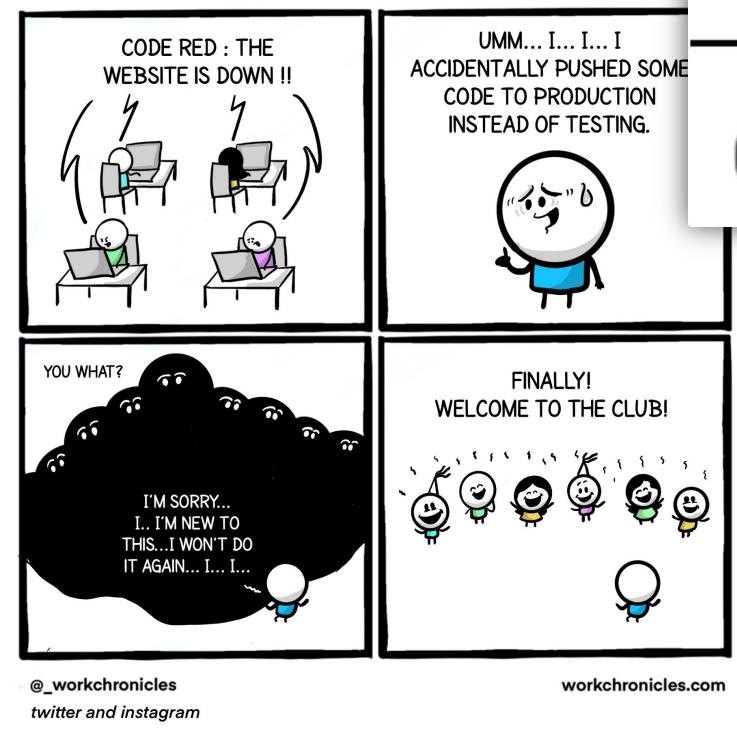
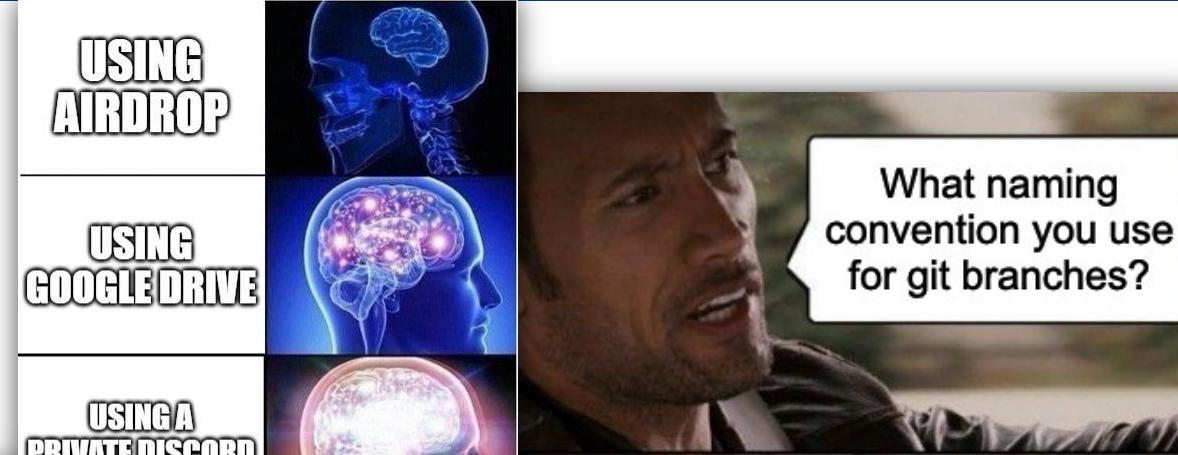
.gitignore



Very Useful Link: <https://ohshitgit.com/en>



- [1] - Official project web: <https://git-scm.com>
- [2] - Free book: <https://git-scm.com/book/en/v2>
- [3] - Very nice and complex tutorial: <https://www.atlassian.com/git/tutorials>
- [4] - Online course: <https://www.udacity.com/course/version-control-with-git--ud123>
- [5] - Plenty of YouTube video tutorial ...



IN CASE OF FIRE 🔥

1. git commit
2. git push
3. git out!



Adam Ligocki

adam.ligocki@vutbr.cz

Brno University of Technology
Faculty of Electrical Engineering and Communication
Department of Control and Instrumentation



Robotics and AI
Research Group