



3 – GIT, the Version Control System

Robotics and Computer Vision (BPC-PRP)

Couse supervisor: Ing. Adam Ligocki, Ph.D.

Slides:Ing. Adam Ligocki, Ph.D.





Robotics and AI

Profile



Ing. Adam Ligocki, Ph.D.

Position: Research Staff

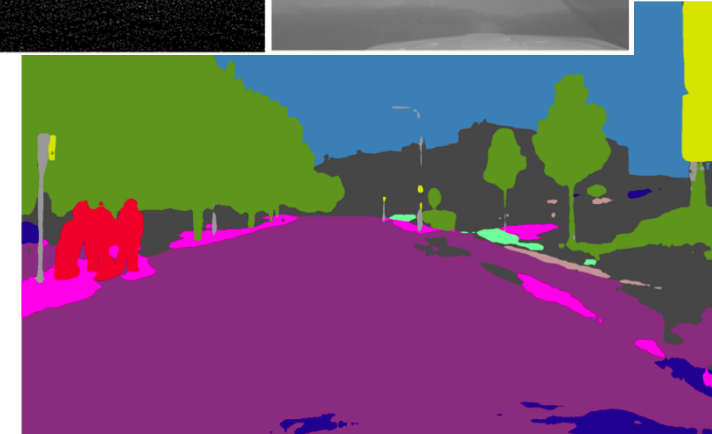
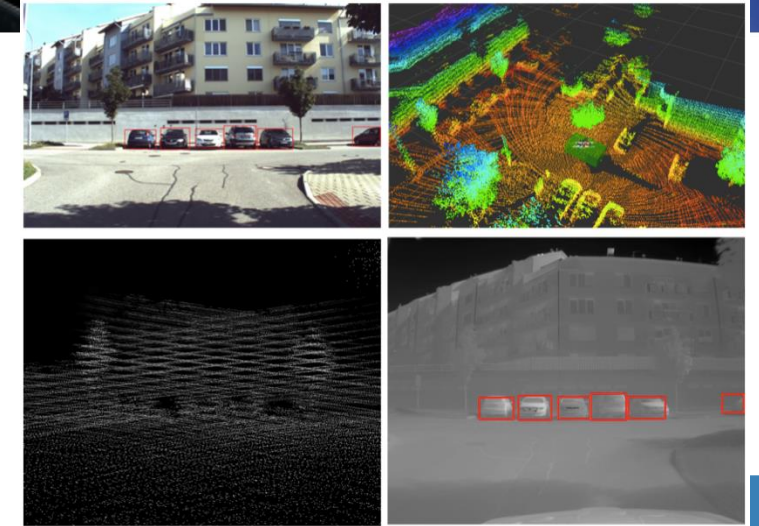
Research: Data Fusion

Room: SE1.102

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

Background:

- Artificial Intelligence
- Neural Networks
- Software Development





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).

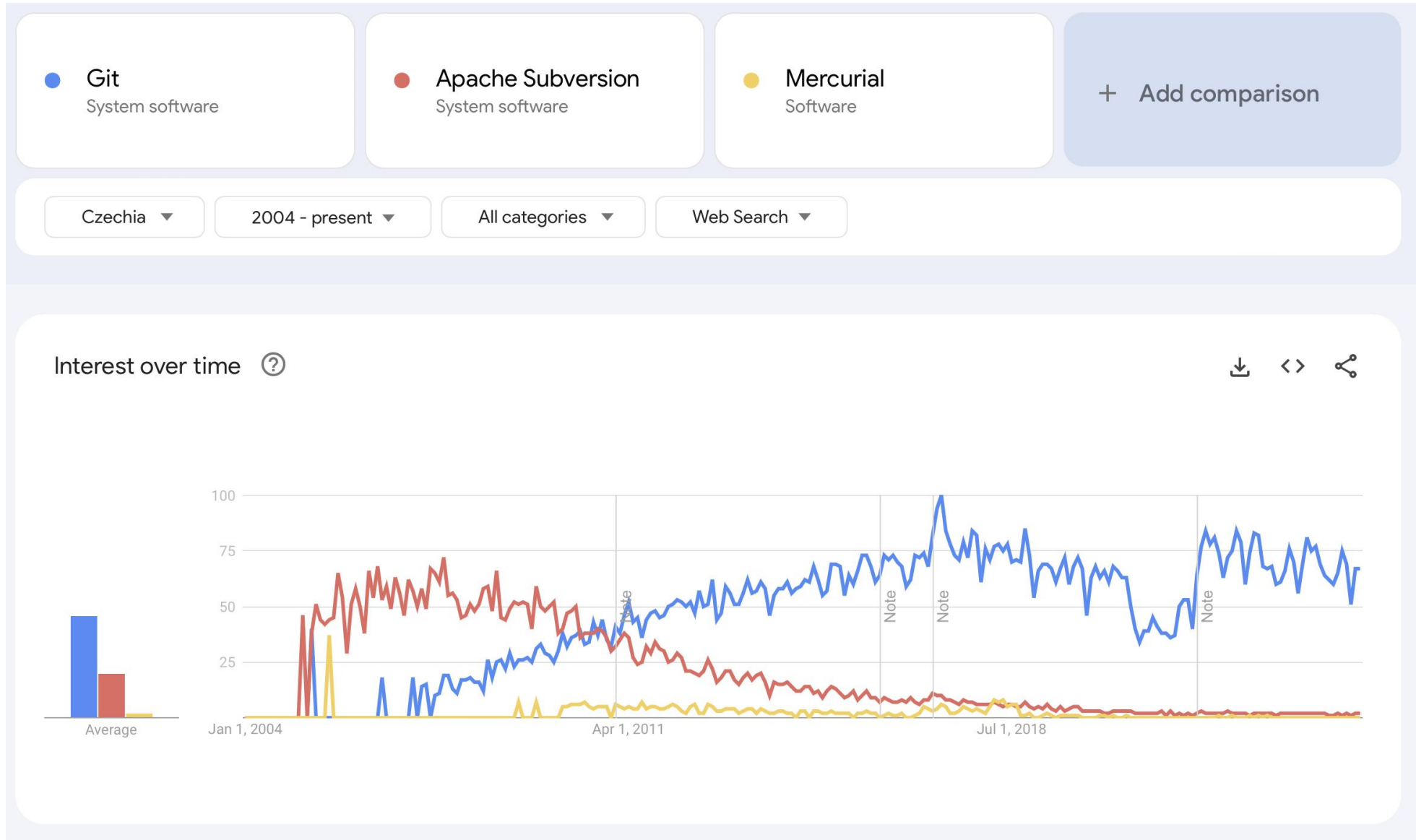
Keeps backup of the project.

Allows multiple developers collaboration.





Why does GIT matter?





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



GIT Basic Terms - Repo

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).

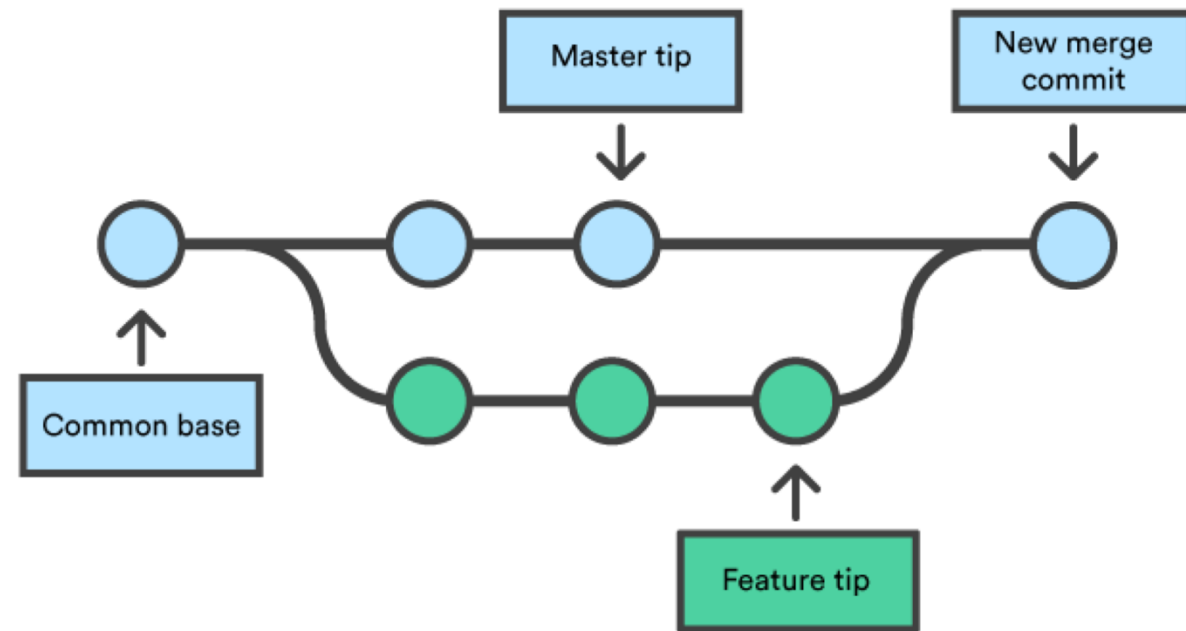
Pushing is the act of uploading your local commits to a remote repository, updating it with your changes.



GIT Basic Terms - Branch

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.



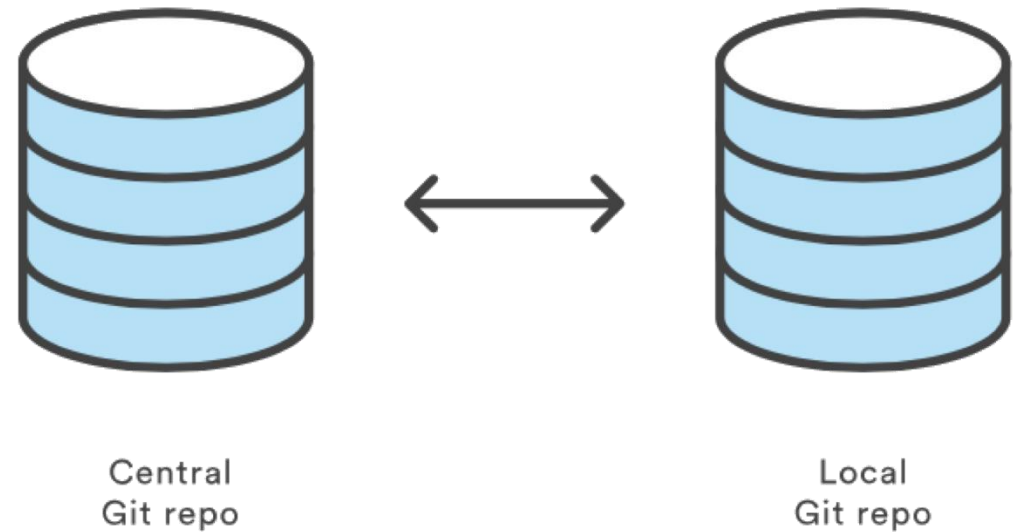
<https://www.atlassian.com/git/tutorials>



GIT – Basic Idea

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.

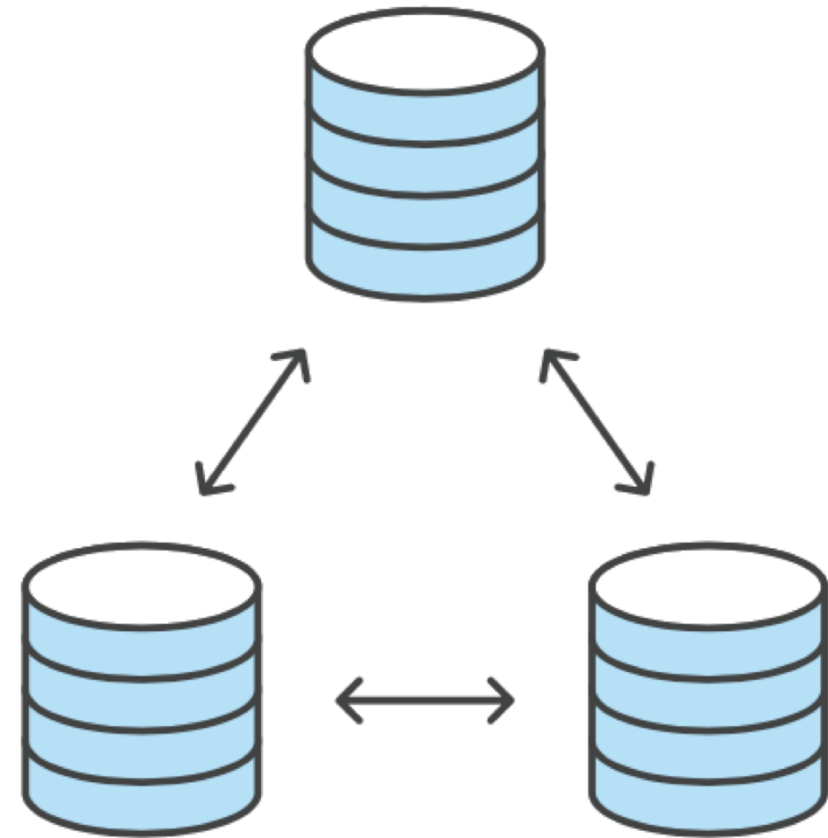




GIT – Distributed System

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>

The screenshot shows a code editor with a project named 'git_ignore_project'. The file explorer on the left shows the project structure, including 'cmake-build-debug', '.gitignore', 'CMakeLists.txt', and 'main.cpp'. The editor window shows the content of the '.gitignore' file, which contains the following entries:

```
1 **/cmake-build-*
2 **/build
3
4 **/.idea
```

The terminal window at the bottom shows the output of the following commands:

```
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 – Large Files

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)

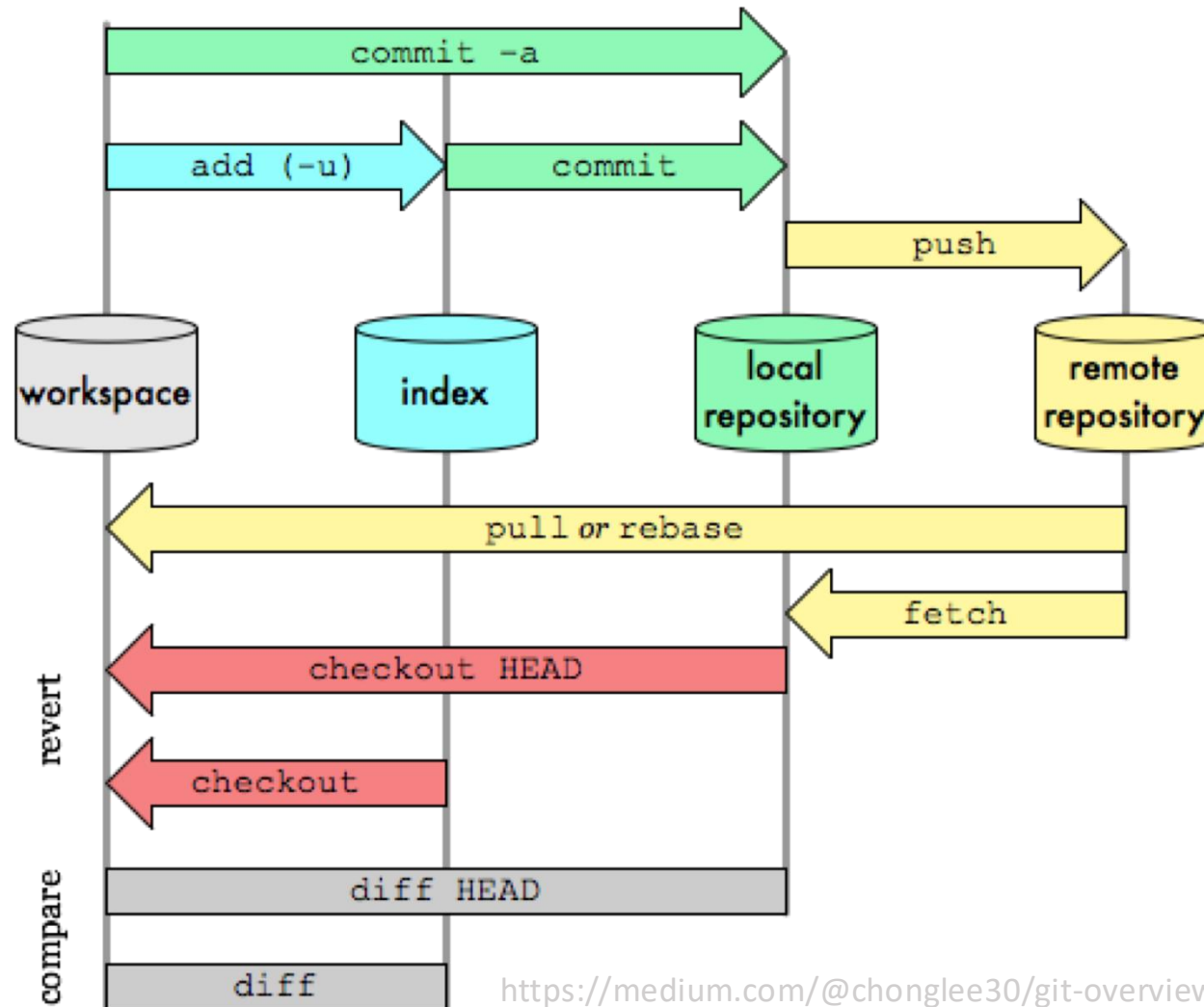


Minecraft Wiki



Git Data Transport Commands

<http://osteele.com>





GIT - Intallation

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>

git --local-branching-on-the-cheap

Search entire site...

About
Documentation
Downloads
GUI Clients
Logos
Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloads

Older releases are available and the [Git source repository](#) is on GitHub.

Latest source Release
2.25.1
Release Notes (2020-02-17)
Download 2.23.0 for Mac

Older releases are available and the [Git source repository](#) is on GitHub.

GUI Clients

Git comes with built-in GUI tools ([git-gui](#), [gitk](#)), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).



GIT – Commands Overview

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 – Commands Overview

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

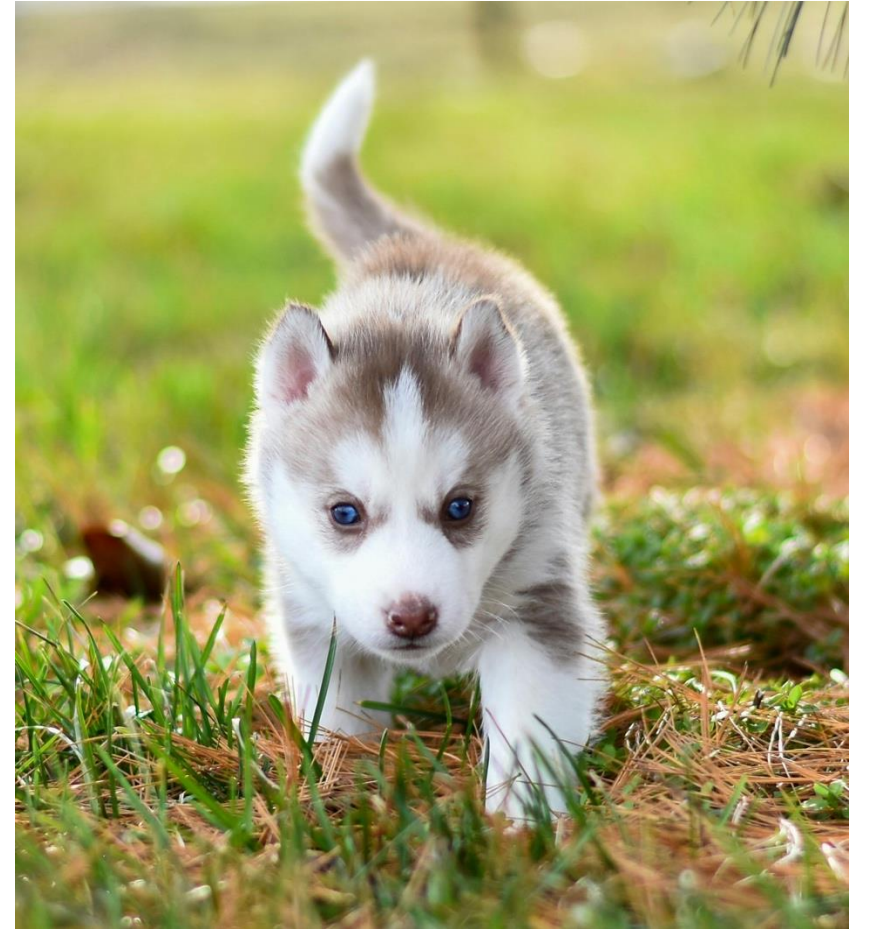
git log - simple visualization of the repository history

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() {
    return 0;
}
```



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



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



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





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

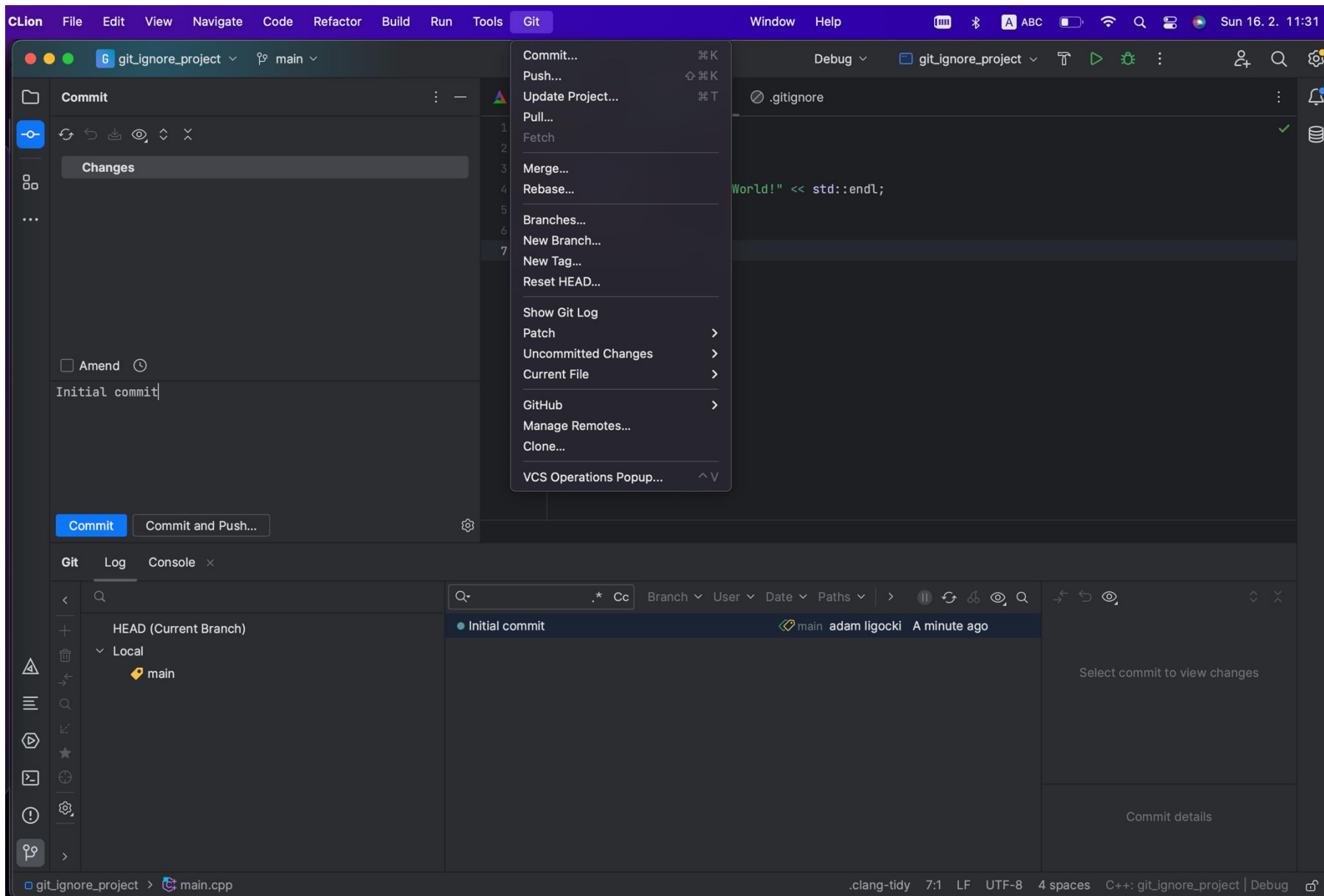
The screenshot displays the CLion IDE interface with the following components:

- Top Bar:** Shows the project name 'git_ignore_project' and the current branch 'main'. It also includes icons for Debug, Run, and other development tools.
- Left Sidebar:** Contains the 'Commit' window. The 'Changes' section lists three files: '.gitignore', 'CMakeLists.txt', and 'main.cpp'. The 'Commit' button is visible at the bottom of this window.
- Main Editor:** Displays the 'main.cpp' file with the following C++ code:

```
1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello, World!" << std::endl;
5     return 0;
6 }
7
```
- Bottom Panel:** Shows the 'Git' tool window. It includes a search bar, a list of branches (HEAD (Current Branch), Local, main), and a message stating 'No changes committed. Commit local changes (⌘K)'. There are also buttons for 'Select commit to view changes' and 'Commit details'.



GIT – VCS integration in CLion





Git Cheat Sheet

For more awesome cheat sheets
visit 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

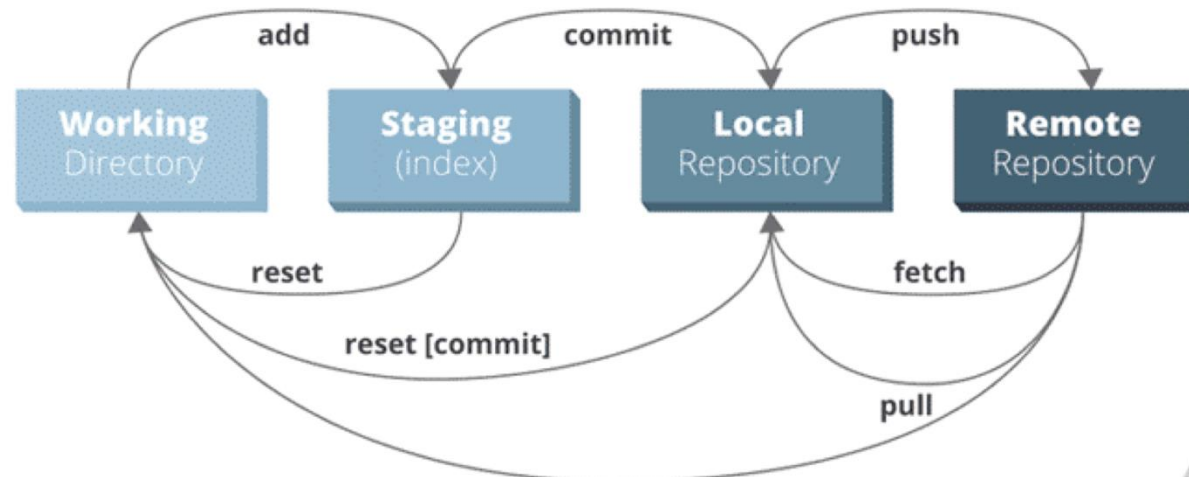
```
$ git push
```

Finally!

When in doubt, use git help

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

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





git init

git checkout

git add

git push

git commit

git pull

git status

.gitignore



GIT - Help



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 ...



GIT – Memes



USING
AIRDROP

USING
GOOGLE DRIVE

USING A
PRIVATE DISCORD

USING
GITHUB



IN CASE OF FIRE



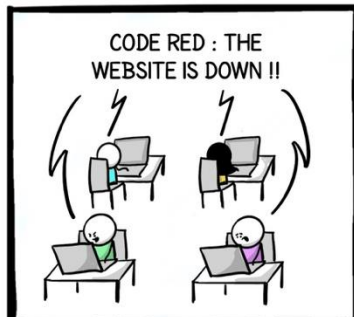
1. git commit



2. git push



3. git out!



UMM... I... I... I
ACCIDENTALLY PUSHED SOME
CODE TO PRODUCTION
INSTEAD OF TESTING.



YOU WHAT?

I'M SORRY...
I.. I'M NEW TO
THIS... I WON'T DO
IT AGAIN... I... I...

FINALLY!
WELCOME TO THE CLUB!





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