3 – GIT, the Version Control System

Robotics and Computer Vision (BPC-PRP)

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

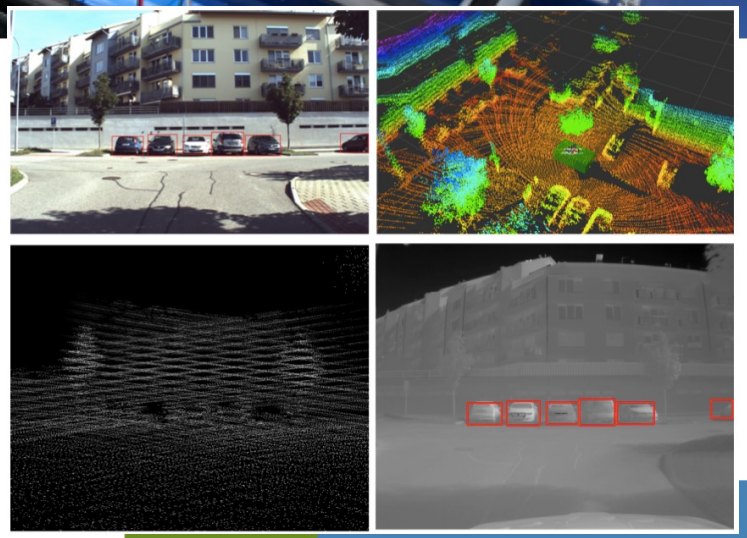
Slides:
Ing. Adam Ligocki, Ph.D.

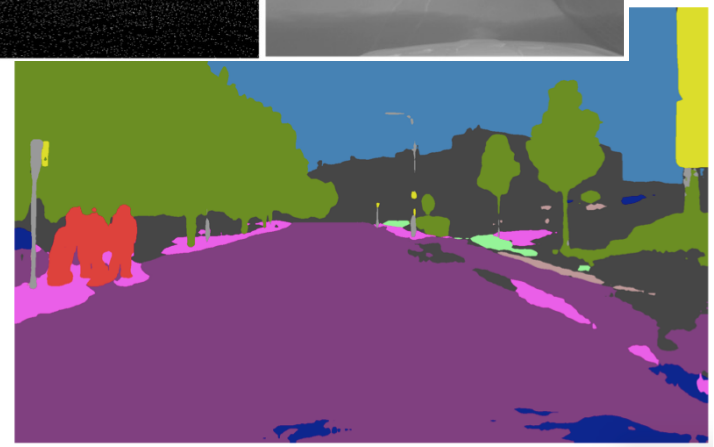
Brno University of Technology
2025

Robotics and AI

Profile

Robotics and AI





Ing. Adam Ligocki, Ph.D.

Position: Research Staff

Research: Data Fusion

Room: SE1.102

Background:

- Artificial Intelligence
- Neural Networks
- Software Development

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

Git – Distributed System for Version Control

GIT Basics

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.

GIT Basic Terms - Commit

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.

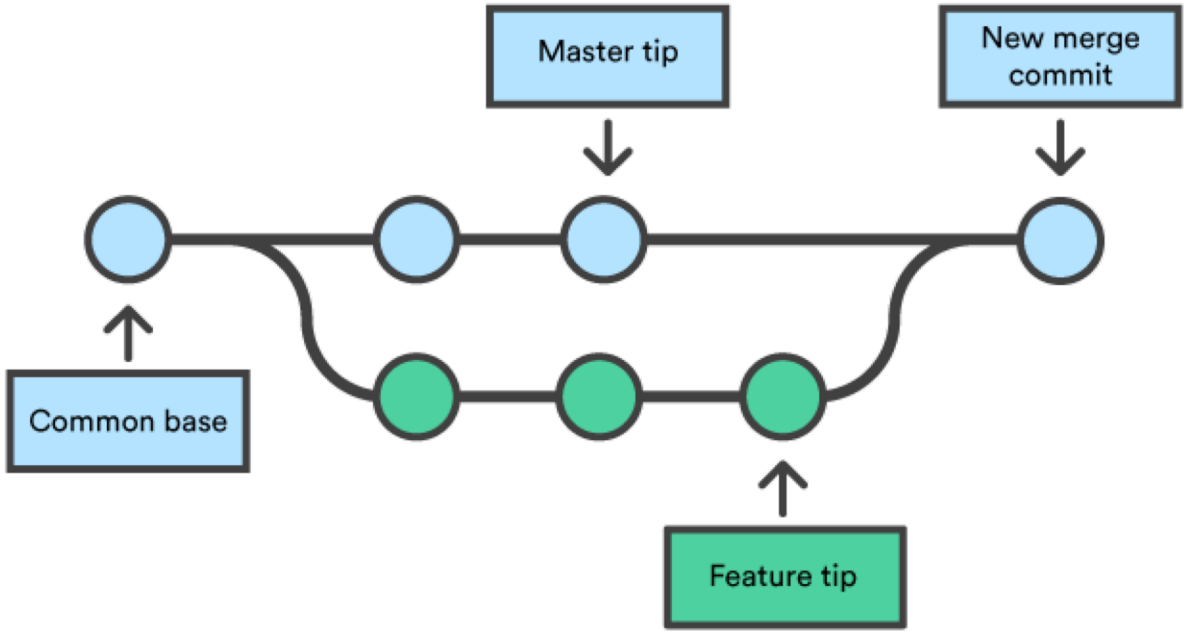
Pulling downloads commits from the remote repository and integrates them into your current branch. Technically, a pull is a combination of a fetch (downloading the commits) followed by a merge (integrating the commits).

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.

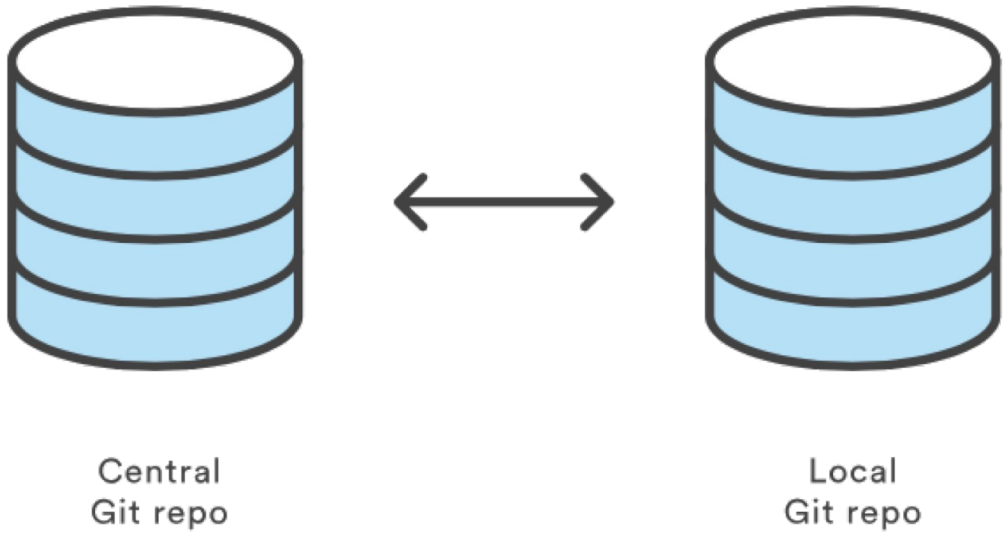


https://www.atlassian.com/git/tutorials

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.

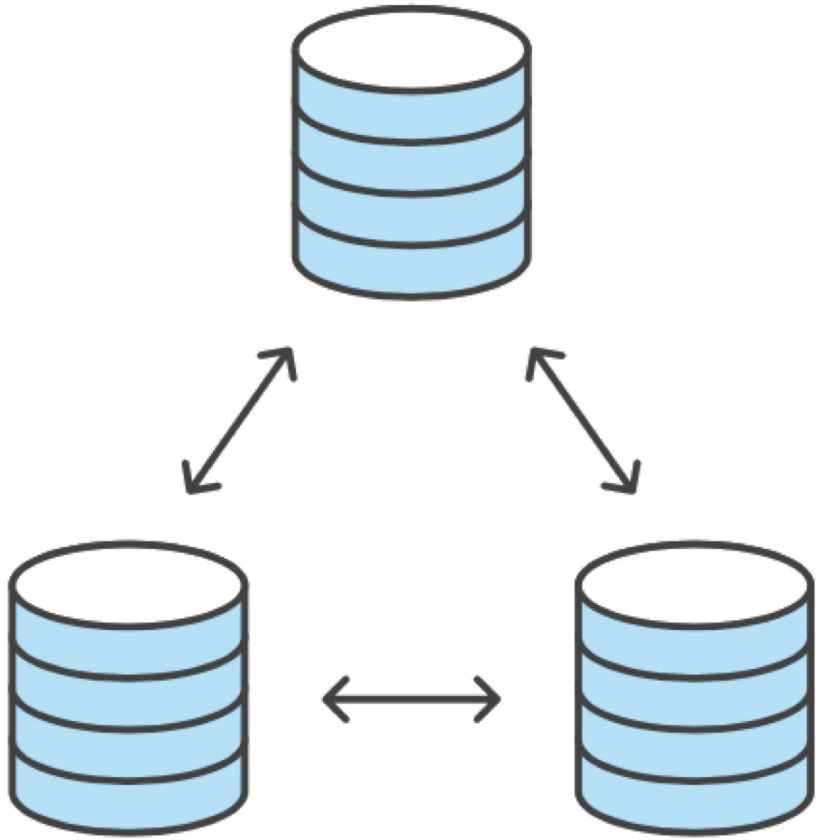


https://www.atlassian.com/git/tutorials

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

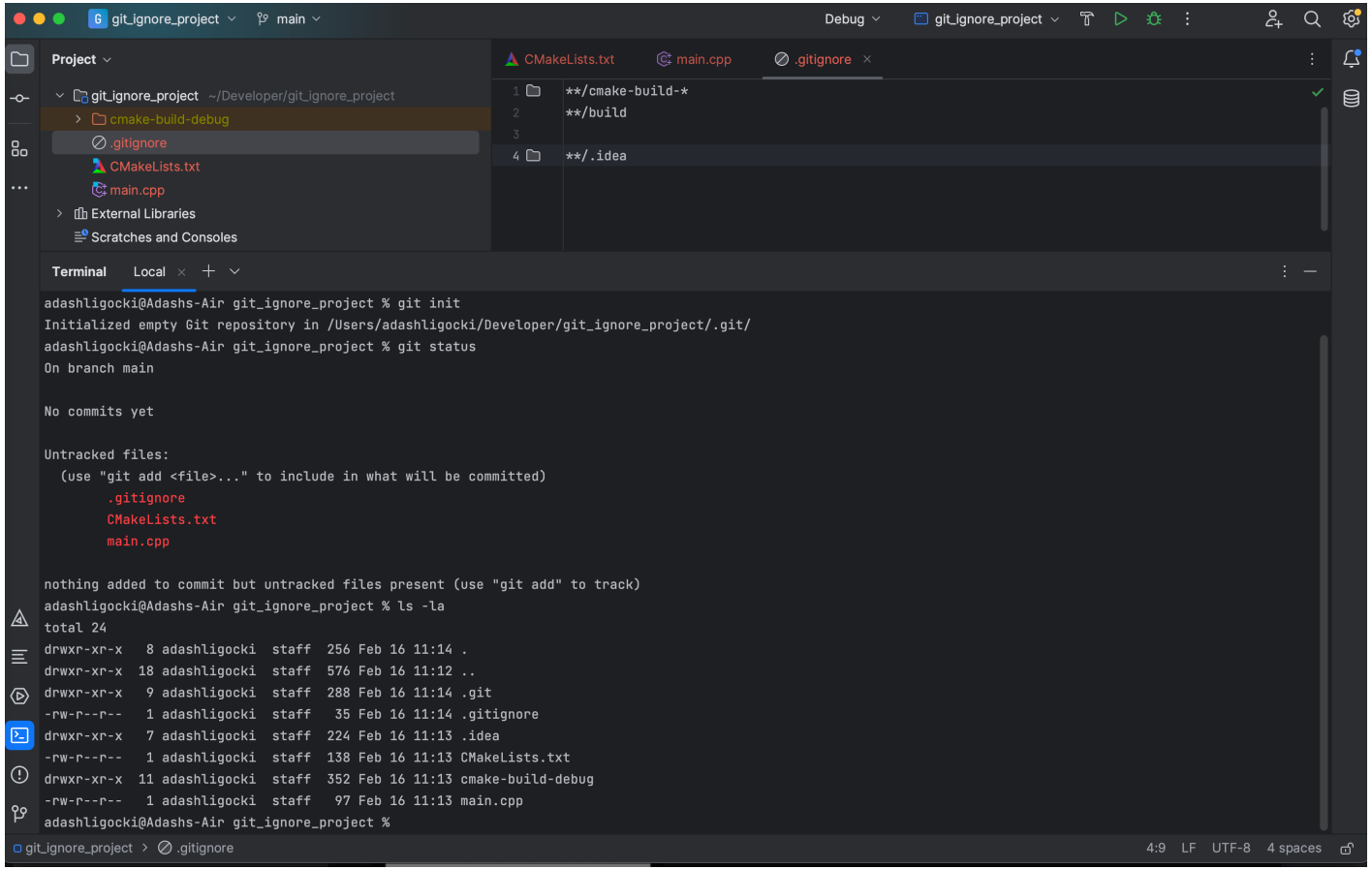
File that defines files or folders that GIT should ignore

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



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)

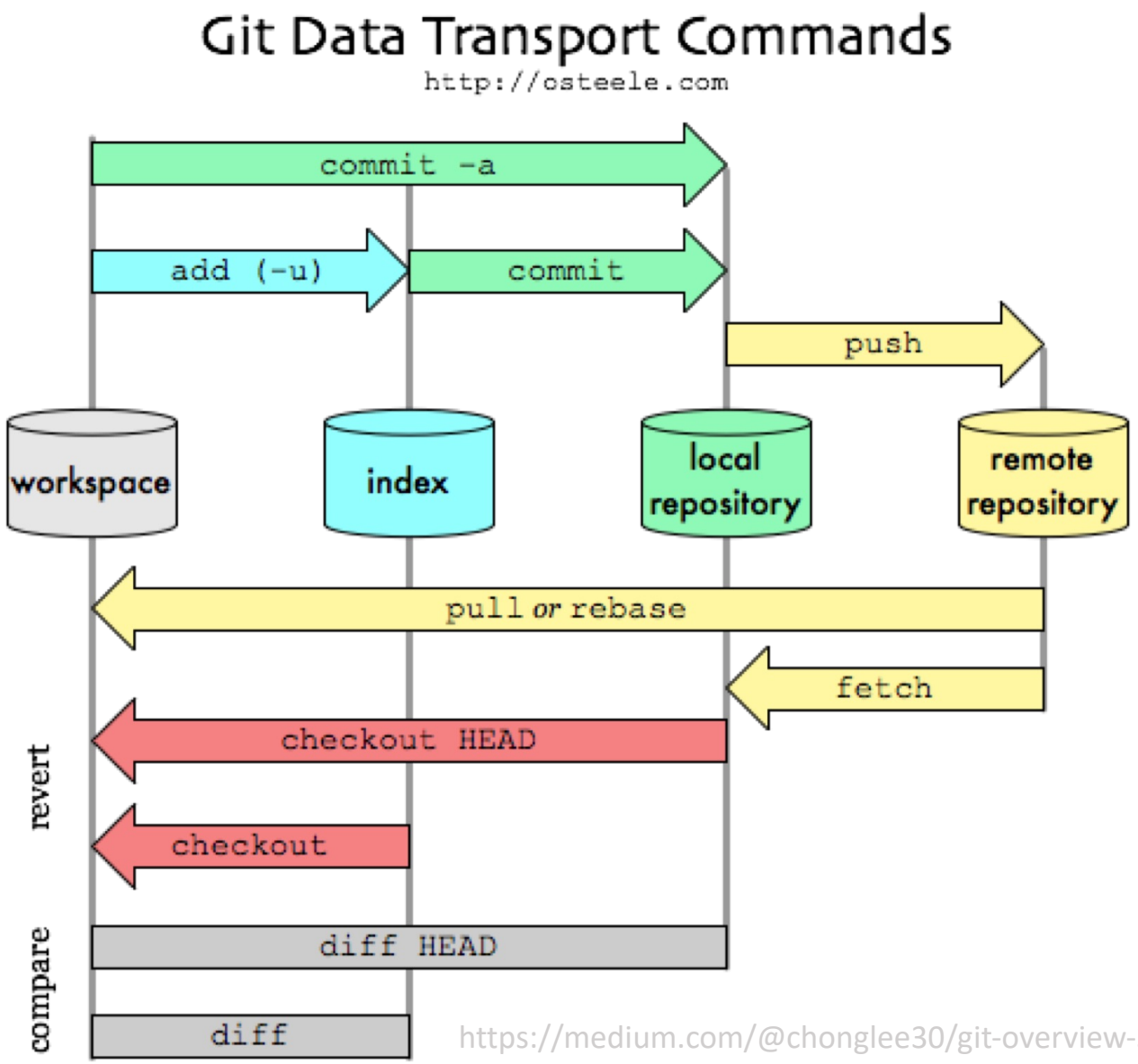
Use **.gitignore** file in the root of the repository to exclude redundant files.

For large files in git, see

Large File Storage (LFS)

Minecraft Wiki

GIT - Workflow



https://medium.com/@chonglee30/git-overview-30618e81eb77

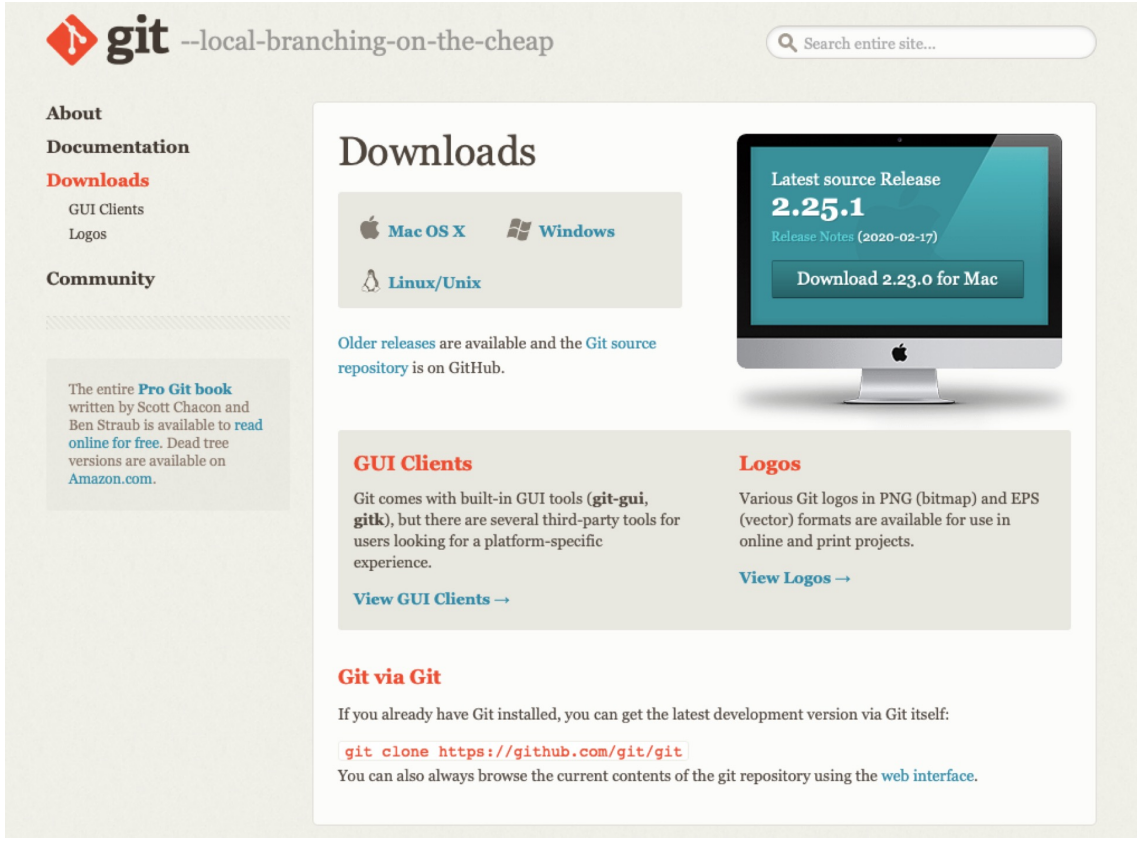
GIT - Intallation

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 – 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 sate
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;
}
```

Why does GIT matter?

Git
System software

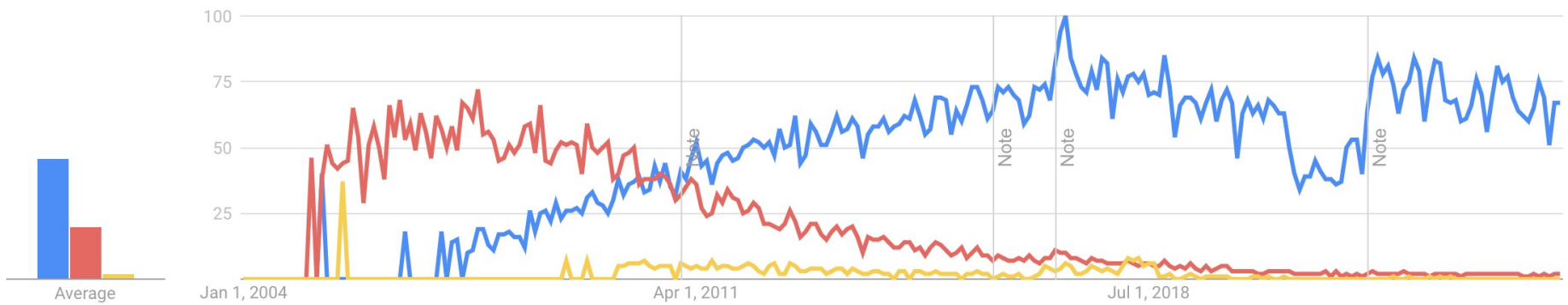
Apache Subversion
System software

Mercurial
Software

+ Add comparison

Czechia | 2004 - present | All categories | Web Search

Interest over time



GIT – Online Platforms and Services

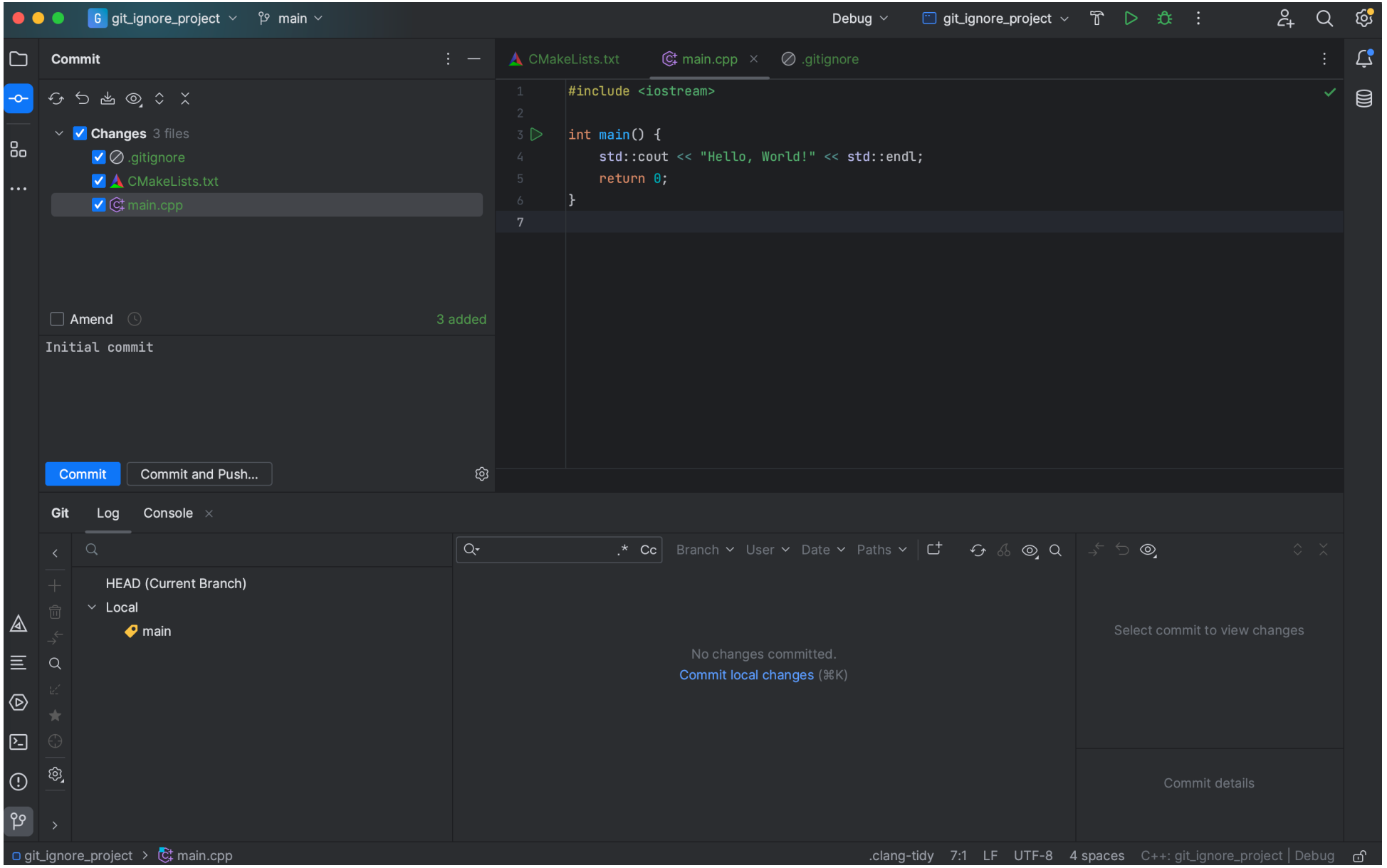
There are several online services
that provides git functionality.

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

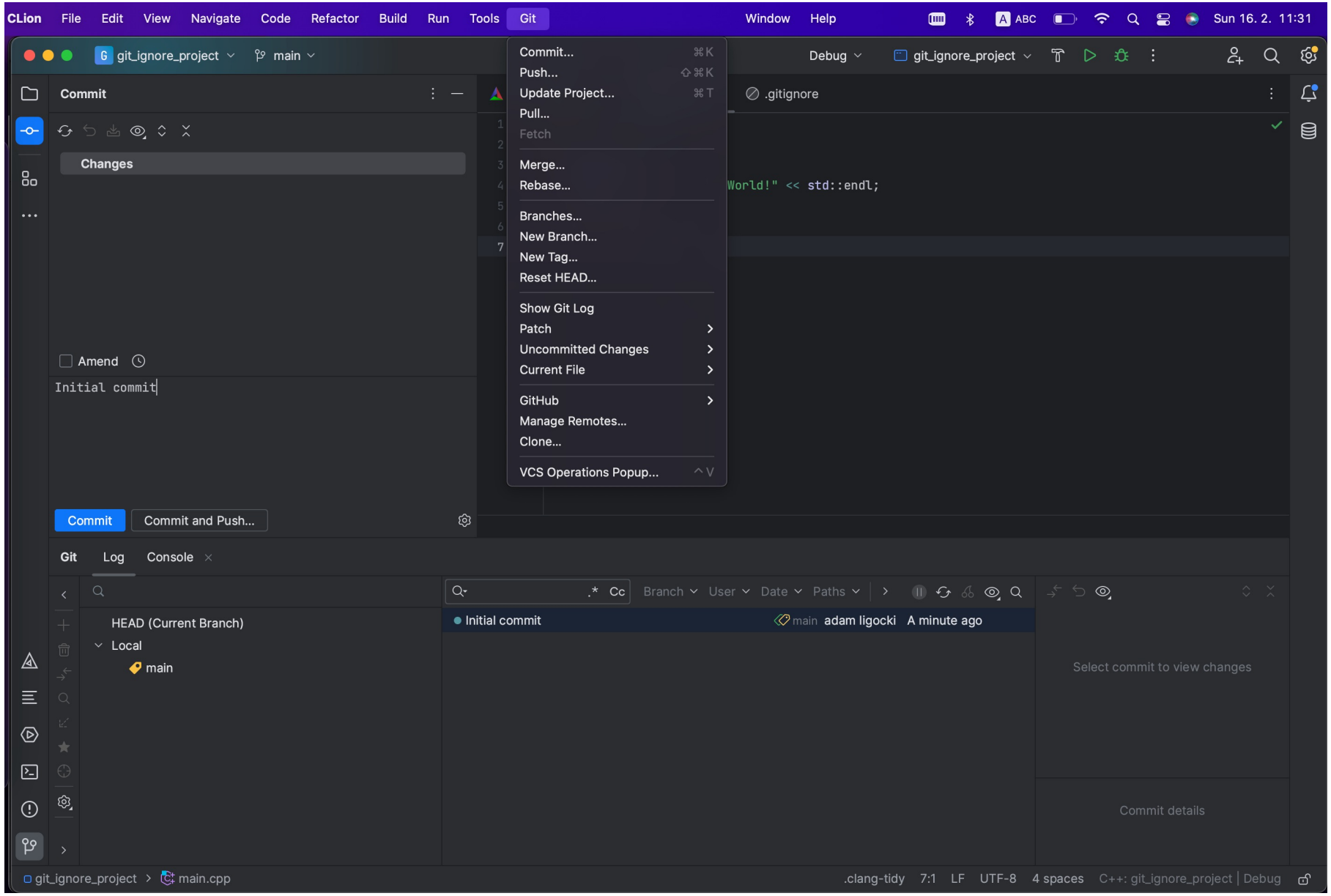
For non-commercial usage, it is free

Can deploy GitLab on your own server

GIT – VCS integration in CLion



GIT – VCS integration in CLion



Git Cheat Sheet

For more awesome cheat sheets visit rebellabs.org!

REBELLABS by ZEROTURNAROUND

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
```

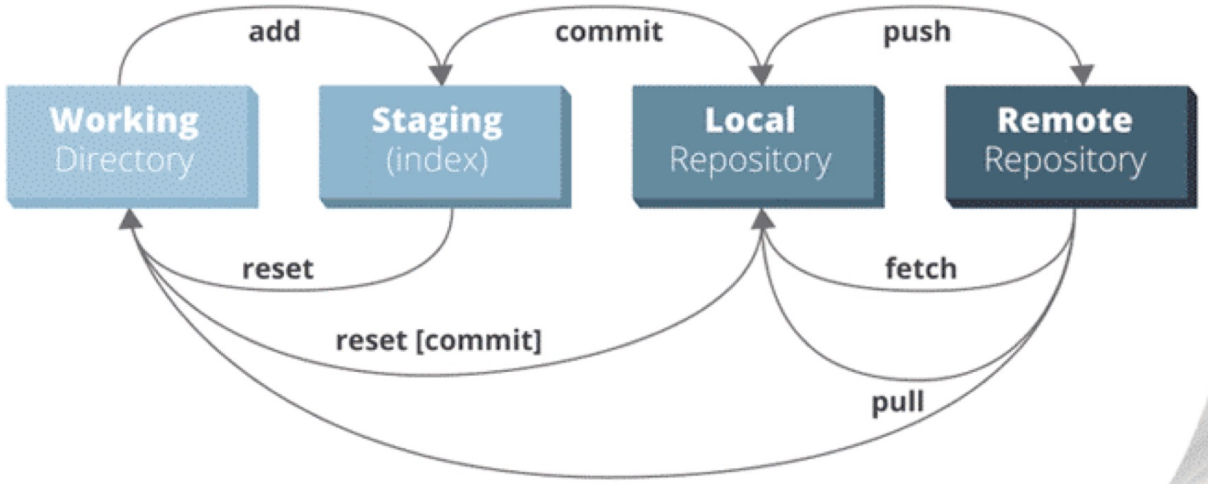
Fetch the latest changes from origin and rebase
```
$ git pull --rebase
```

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.



BROUGHT TO YOU BY JRebel

GIT Minimum

git init

git add

git commit

git status

git checkout

git push

git pull

.gitignore

GIT - Help

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

GIT – Next Step

[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

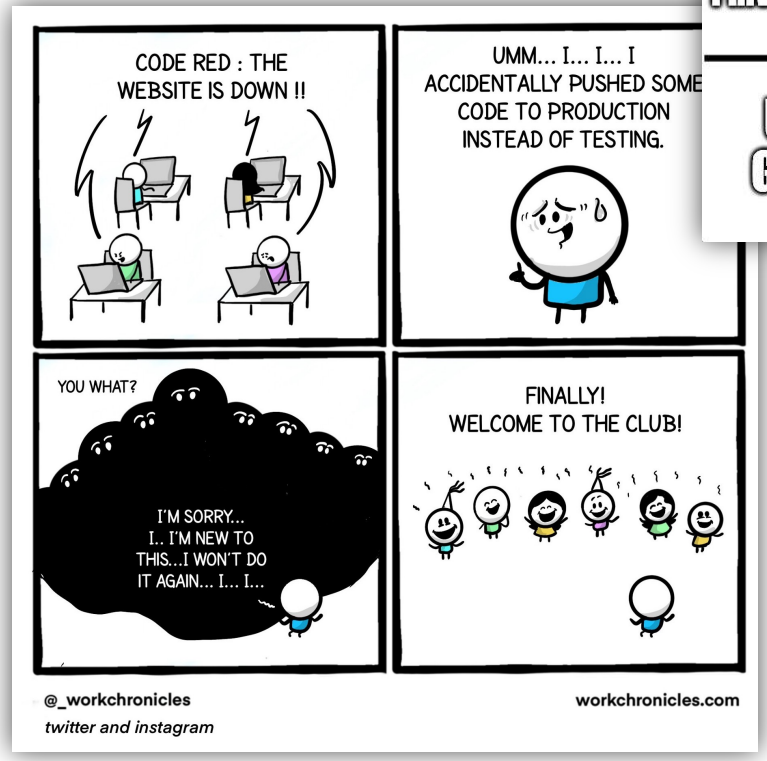
[4] - Online course: https://www.udacity.com/course/version-control-with-git--ud123

[5] - Planty of YouTube video tutorial …

GIT – Memes


ONE DOES NOT SIMPLY
UNDERSTAND GIT
MemesHappen


USING AIRDROP
USING GOOGLE DRIVE
USING A PRIVATE DISCORD
USING GITHUB


CODE RED : THE WEBSITE IS DOWN !!
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!
@_workchronicles
twitter and instagram
workchronicles.com


What naming convention you use for git branches?
What's a branch? I only work in master.
imgflip.com


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