Get To Git

Session 1

Agenda

01 Introduction to Open Source

02 Contributing to Open Source

03 What is Git & its Benefits

04 Git Workflow

05 Main Git Commands

About Ahmed Khaled (Akayiz)

- SP Project
- OS Project
- Google Summer of Code
- Random Hobbies

What about you?

Agenda

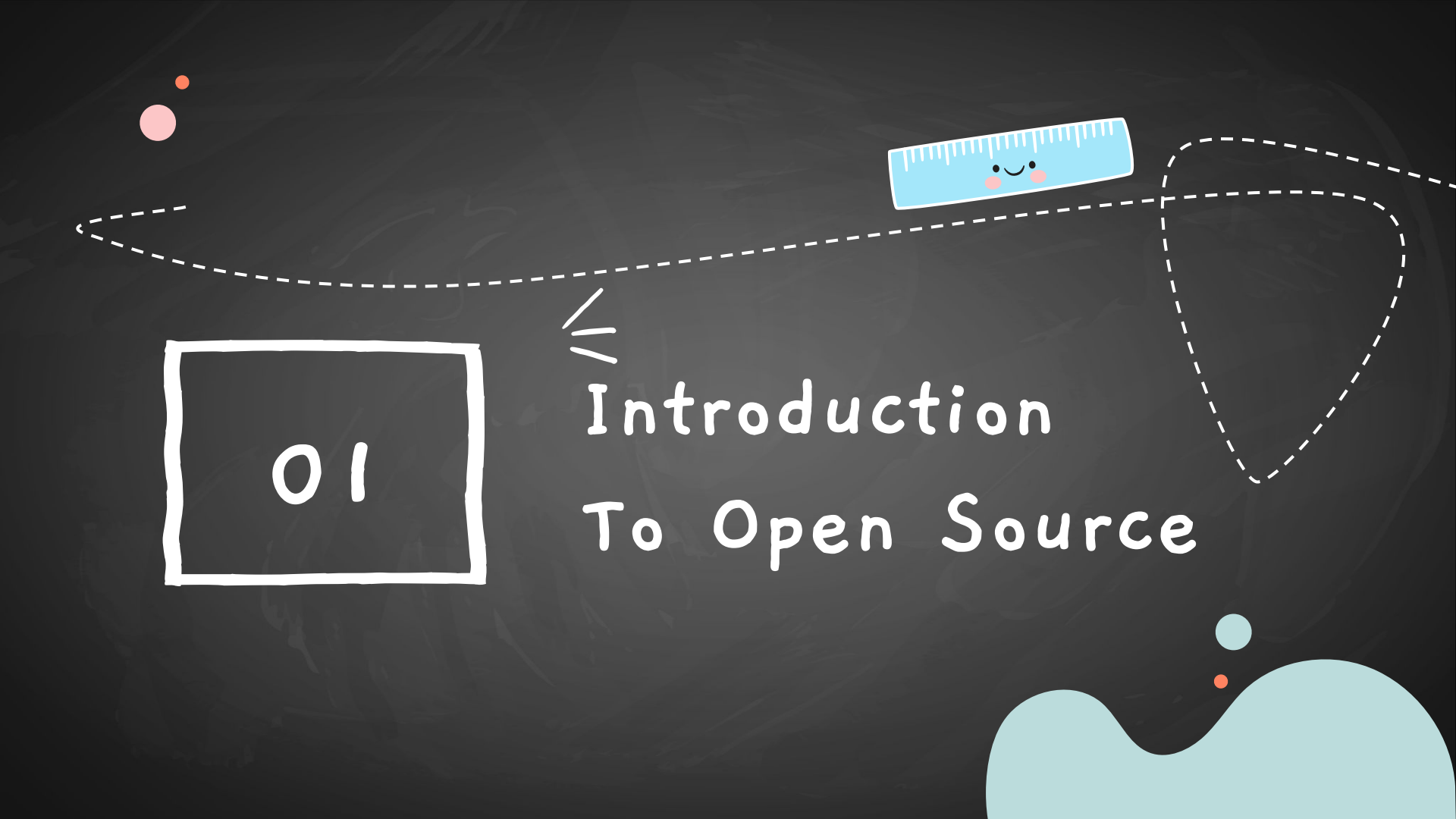
01 Introduction to Open Source

02 Contributing to Open Source

03 What is Git & its Benefits

04 Git Workflow

05 Main Git Commands


01
Introduction
To Open Source

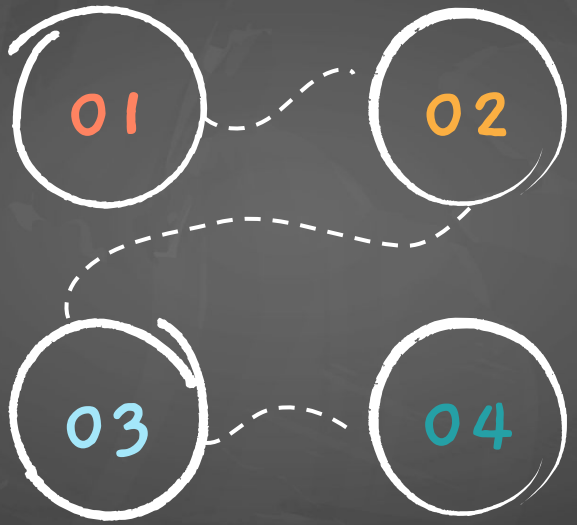
What is Open Source?

There are many definitions, but we will pick the **FOSS** definition!

FOSS ≠ FOS

The 4 freedoms of FOSS

Freedom to run the program as you wish, for any purpose.



Freedom to study how the program works, and modify it.

Freedom to distribute copies so you can help others.

Freedom to distribute modified copies of the program.

Why is FOSS super cool?

01 Transparency

02 Encourages Collaboration

03 Customizability

04 Minimizes Reinventing The Wheel

Agenda

01 Introduction to Open Source ✅

02 Contributing to Open Source

03 What is Git & its Benefits

04 Git Workflow

05 Main Git Commands

02

Contributing To Open Source

Why Should You Contribute?

It teaches you real skills...

- Navigating a code base
- Problem solving
- Working with others
- Documenting your work

It improves the tools you use

- Add a feature
- FIx a bug
- Improve documentation

Why Should You Contribute?

It builds your reputation...

- Showcases your work
- Networking opportunity

It's fun!

- Application process is a lot better.
- A chance to contribute to software that is used by real users.
- You can get very well compensated 💸

2

I'm in!

But **how** can I contribute?

Wait, are you actually?

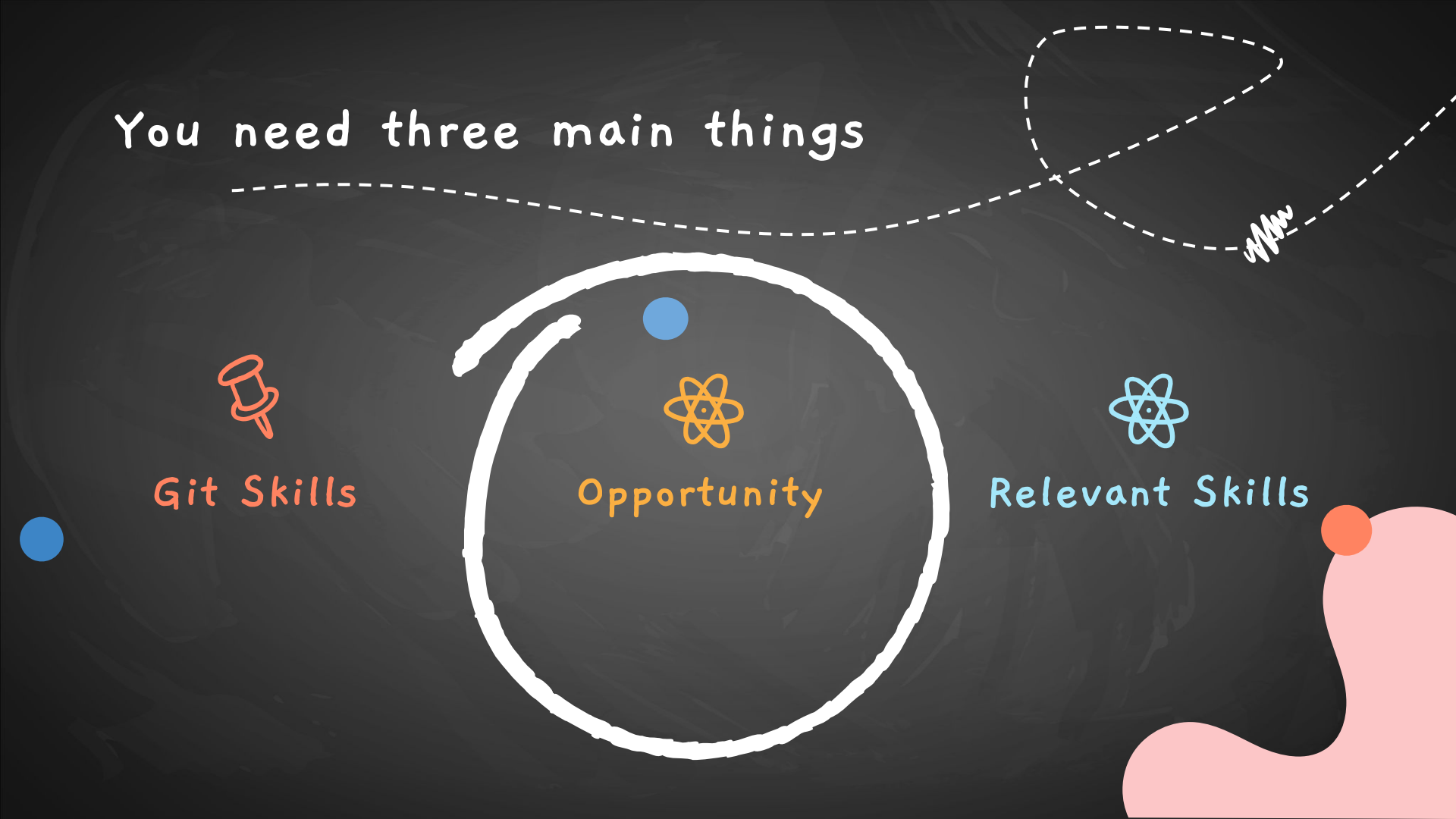
You need three main things

Git Skills

Opportunity

Relevant Skills

You need three main things



Contribution Opportunities

Google Summer of Code

MLH Fellowship

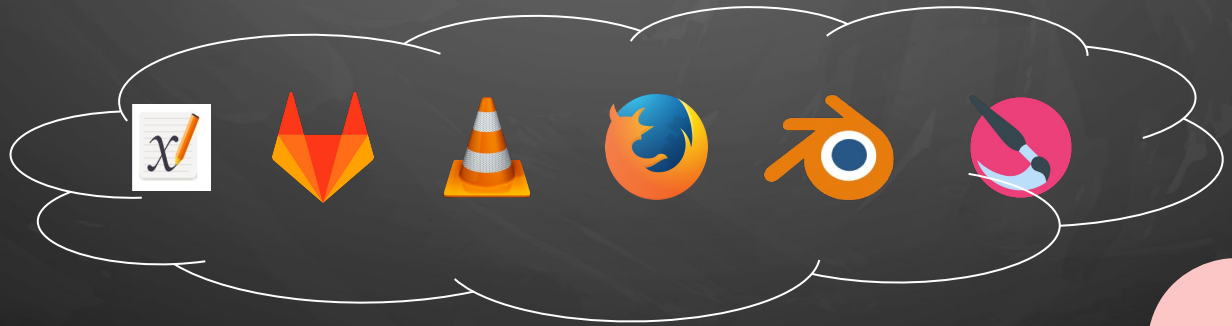
Google Summer of Code

- What is it?
- Benefits
  - Enormous Resume Boost
  - Work on Real & Useful Projects
  - Self-esteem Boost
  - Awesome Compensation
- Application Process?

Get involved in Open Source

Events are not the only way to contribute to open source!

Checkout the issues section in any open source project.



Agenda

01 Introduction to Open Source ✅

02 Contributing to Open Source ✅

03 What is Git & its Benefits

04 Git Workflow

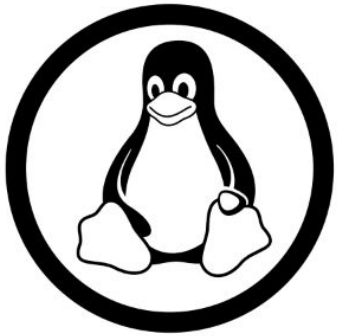
05 Main Git Commands

03

What's Git?

BitKeeper
Scalable Version Control

2002 — **Linux** Team Started Using **BitKeeper** SCM from free version from **BitMover**



2005 — **BitMover** Stopped providing free version

8 April 2005 — **Git** Was Born [first commit]

An important question should be asked 🤔

```
Initial revision of "git", the information manager from hell
```

In this commit, he included a file called `README`. The first paragraph in this file reads:

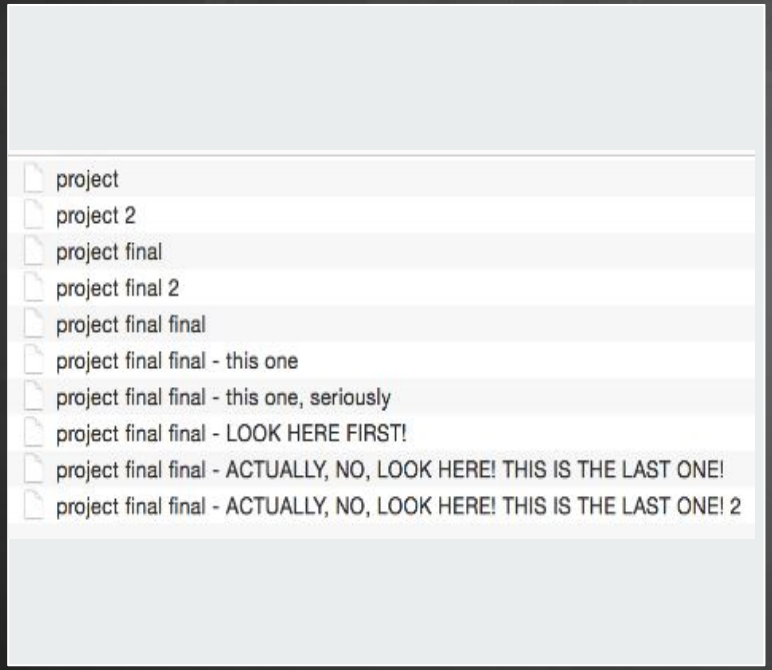
```
GIT - the stupid content tracker

"git" can mean anything, depending on your mood.

 - random three-letter combination that is pronounceable, and not
   actually used by any common UNIX command.  The fact that it is a
   mispronunciation of "get" may or may not be relevant.
 - stupid. contemptible and despicable. simple. Take your pick from the
   dictionary of slang.
 - "global information tracker": you're in a good mood, and it actually
   works for you. Angels sing, and a light suddenly fills the room.
 - "goddamn idiotic truckload of sh*t": when it breaks

This is a stupid (but extremely fast) directory content manager.  It
doesn't do a whole lot, but what it _does_ do is track directory
contents efficiently.
```

Git

Software

Git is a distributed version control software system that is capable of

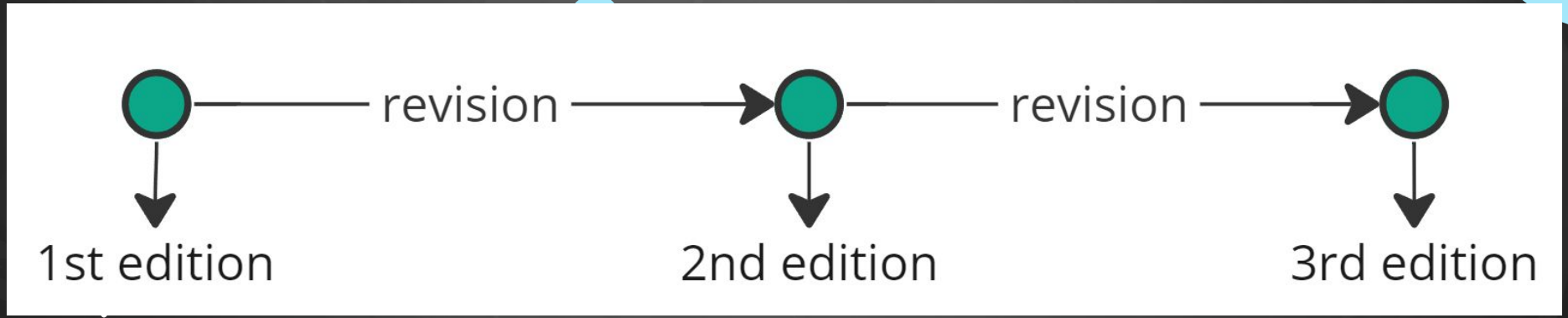
Life Before Version Control



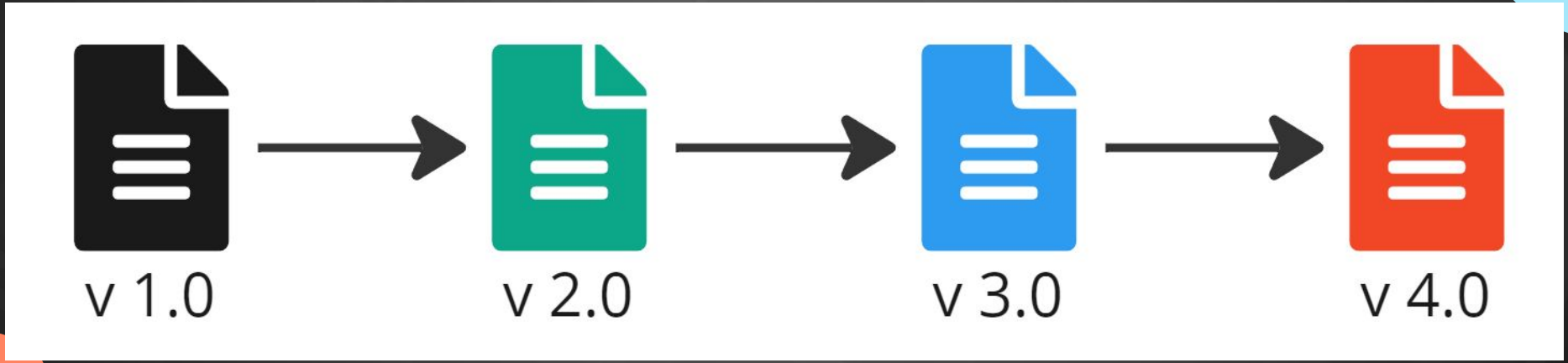
Distributed Version Control System DVCS

Version Control System?

Version



Renaming




SOFTWARE VERSIONS
SOFTWARE VERSIONS
EVERYWHERE
makeameme.org

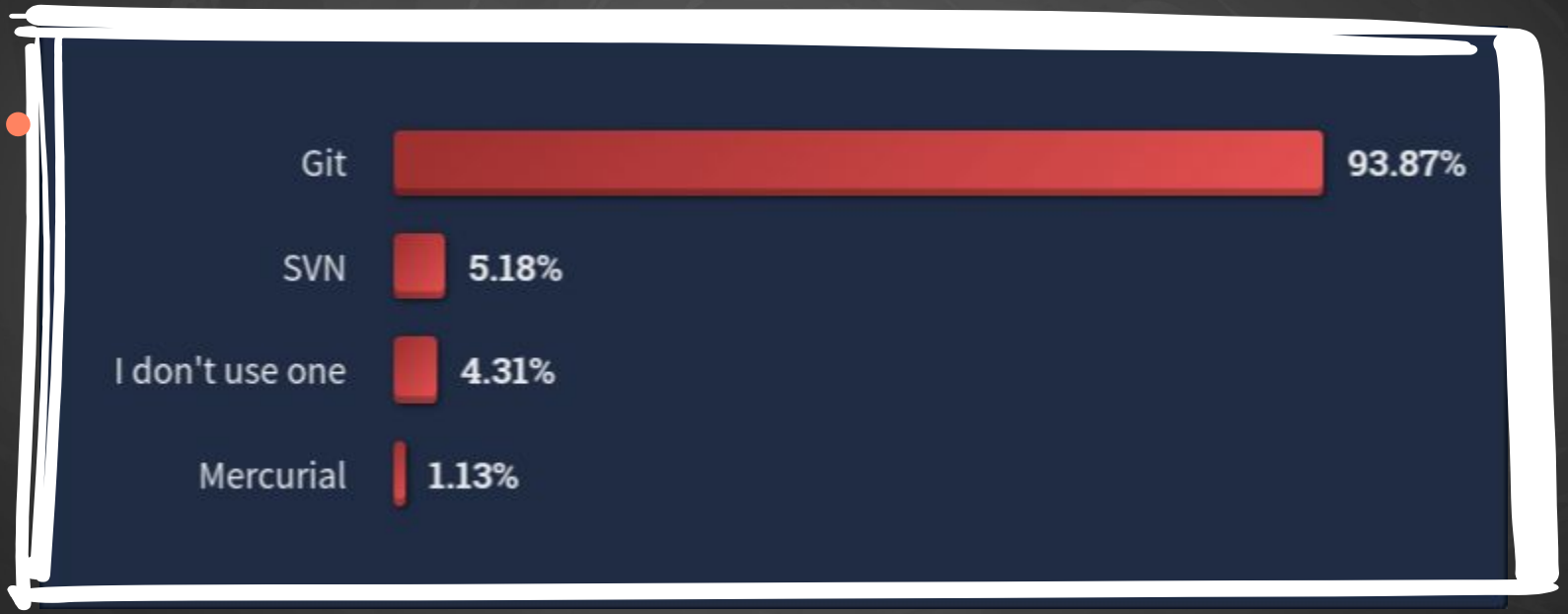
Version Control System

Keep track of all changes

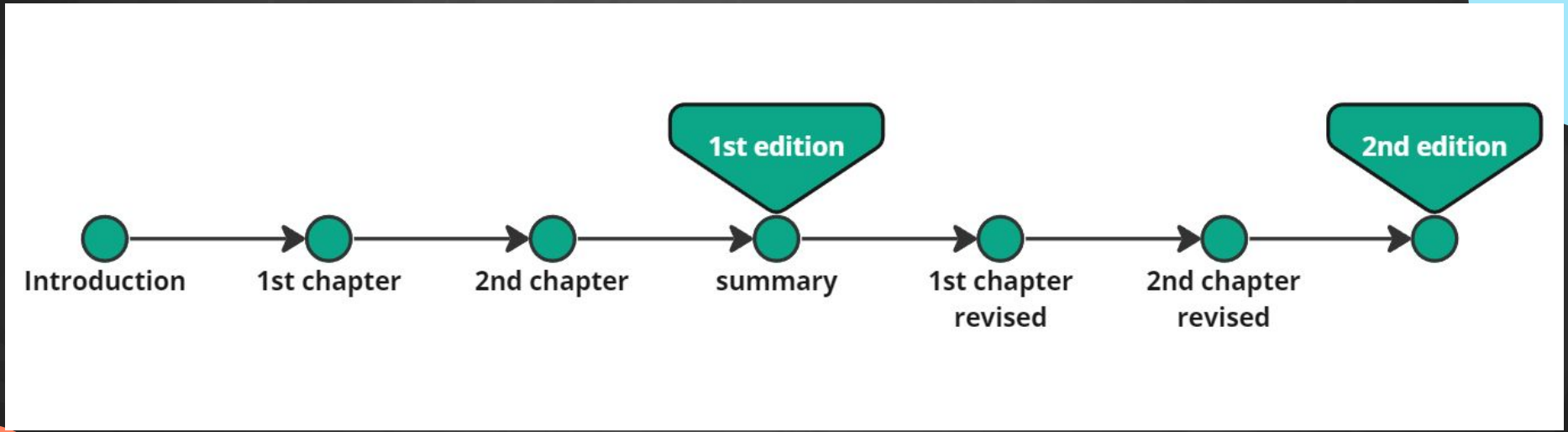
- Allows people to work simultaneously

- Examples:
  - Git
  - Mercurial
  - Subversion
  - Perforce
  - Jujutsu

The Primary VCSs Used By Developers*



* Stack Overflow 2022 Developer Survey


1st edition
2nd edition
Introduction
1st chapter
2nd chapter
summary
1st chapter revised
2nd chapter revised

Distributed Version Control System DVCS

Distributed ?

Distributed Vs Centralized VCS

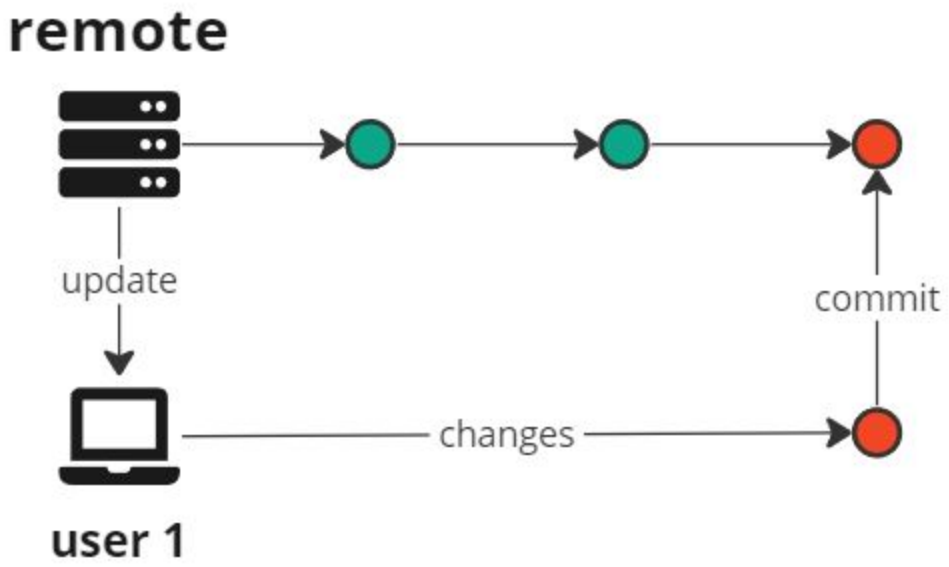
Centralized CVCS

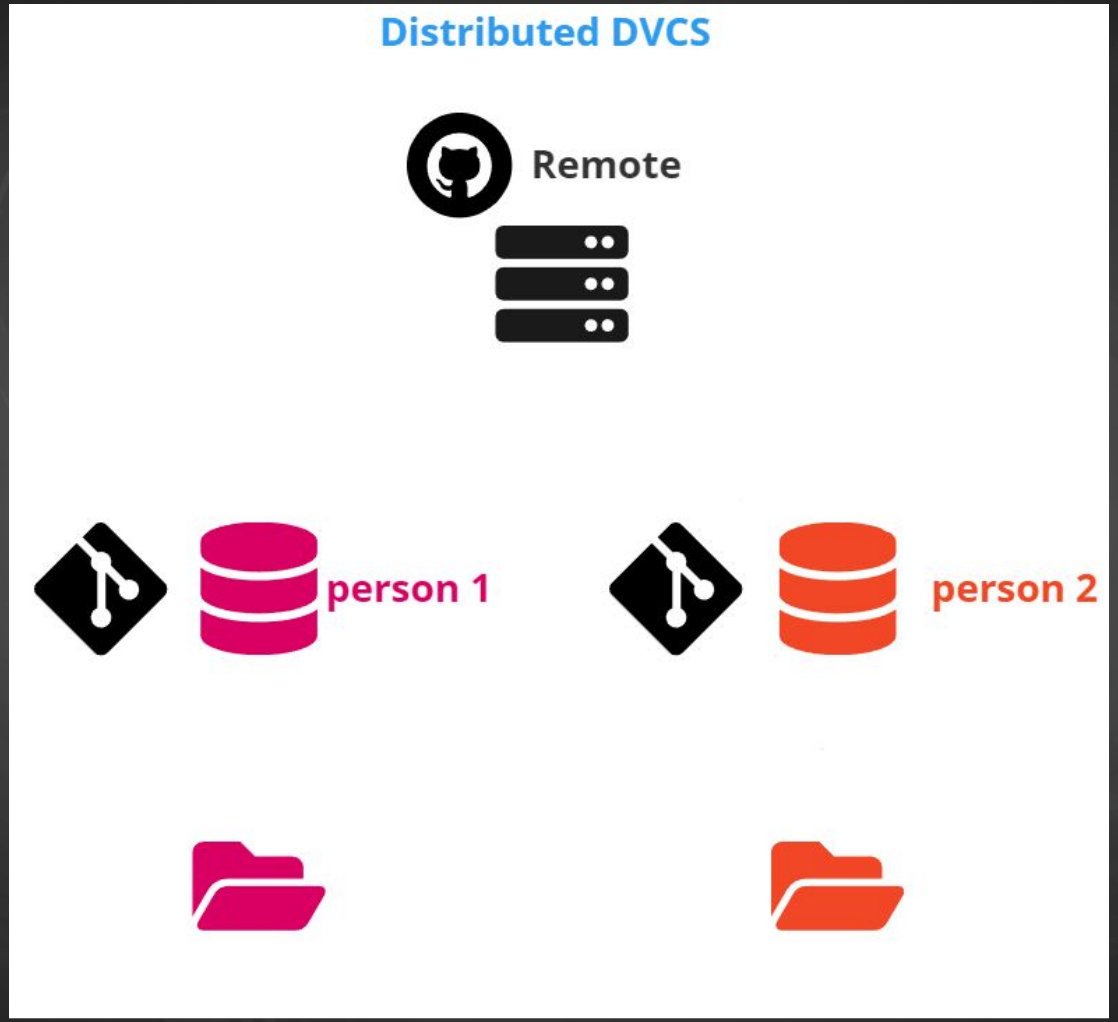
Remote

person 1

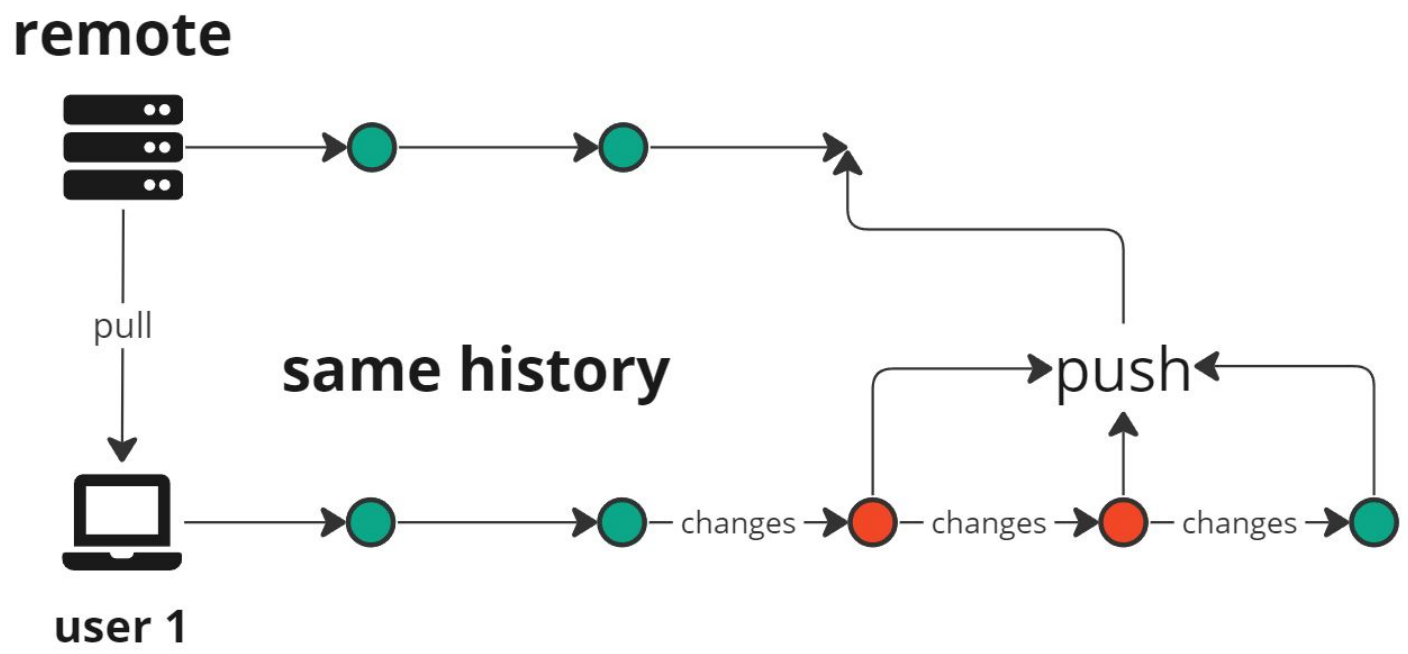
person 2

CVCS




Distributed DVCS
Remote
person 1
person 2

DVCS


remote
pull
same history
push
changes
changes
changes
user 1

DVCS Pros

- Better commits
- Allows offline work

DVCS Cons

- Has a small extra learning curve

Agenda

01 Introduction to Open Source ✅

02 Contributing to Open Source ✅

03 What is Git & its Benefits ✅

04 Git Workflow

05 Main Git Commands

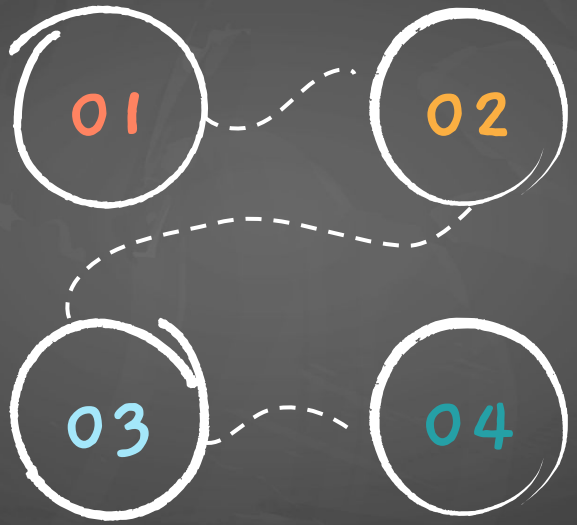
04

Git Workflow

Git Workflow

Working Directory

Staging Area (Index)

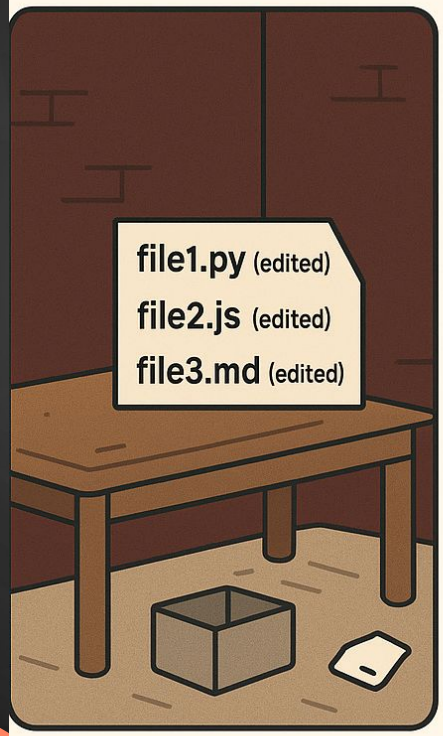


Commit History

Branches

Next session!

WORKING DIRECTORY



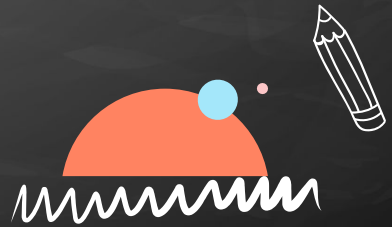
STAGING AREA

COMMIT HISTORY

Working Directory

It is the place where you actually edit files, compile code, and develop your project.

The working directory is a normal folder, except that you now have access to Git commands that can manage the content of that folder.

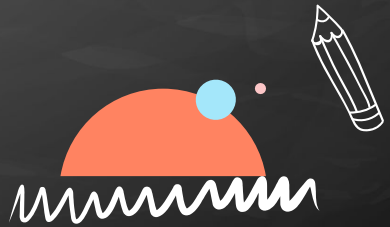


Staging Area (Index)

It is an intermediary point between the working directory and the project history.

Instead of forcing you to commit all changes at once, Git lets you group them into related sets of changes.

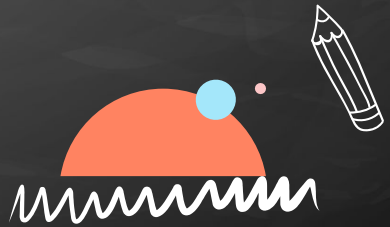
Note: staged changes are not yet part of your history.



Commit History

This contains changes that have been recognized by Git as checkpoints in the project history.

These changes were first added to the staging area and then **committed** .



Agenda

01 Introduction to Open Source ✅

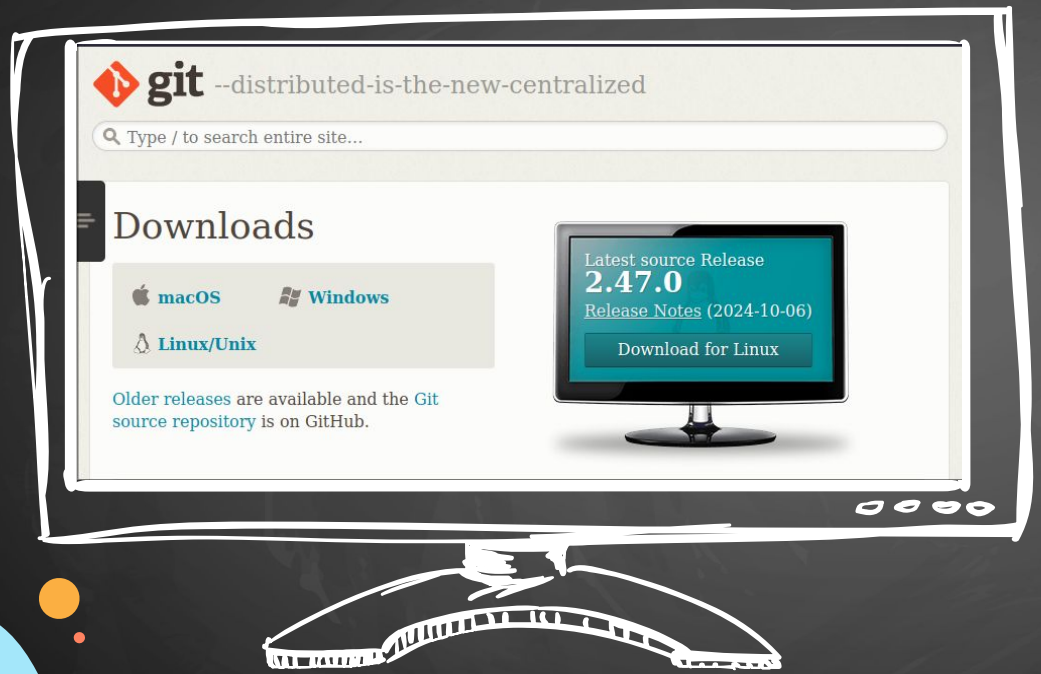
02 Contributing to Open Source ✅

03 What is Git & its Benefits ✅

04 Git Workflow ✅

05 Main Git Commands

Installing and Configuring Git



Installation

Configuration

After installing Git, run the following commands.

```
git config --global user.name <your name>

git config --global user.email <your email>
```

05

Main Git Commands

Main Git Commands

- git init
- git add
- git status
- git commit

nice background
WisdomCasual committed on Dec 12, 2023 e7141d2

register look changed
AymanM04 committed on Dec 12, 2023 51784ea

transition is transitioning
KareemEssam7 committed on Dec 12, 2023 ecf36ce

last working version
KareemEssam7 committed on Dec 12, 2023 c9fe806

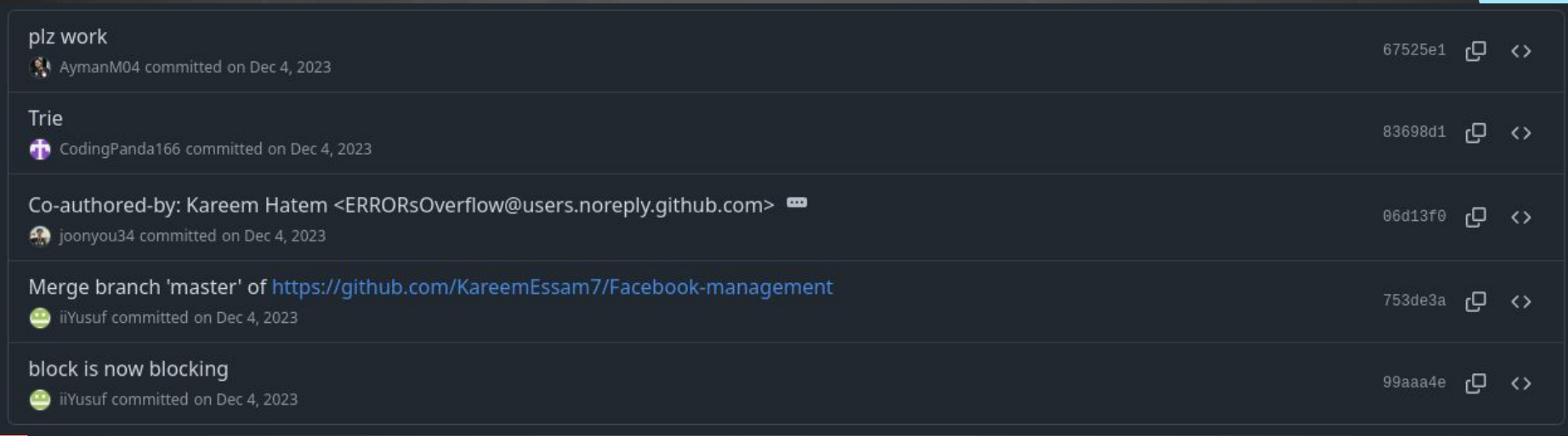
Merge branch 'master' of https://github.com/KareemEssam7/Facebook-management
WisdomCasual committed on Dec 12, 2023 7d5d167

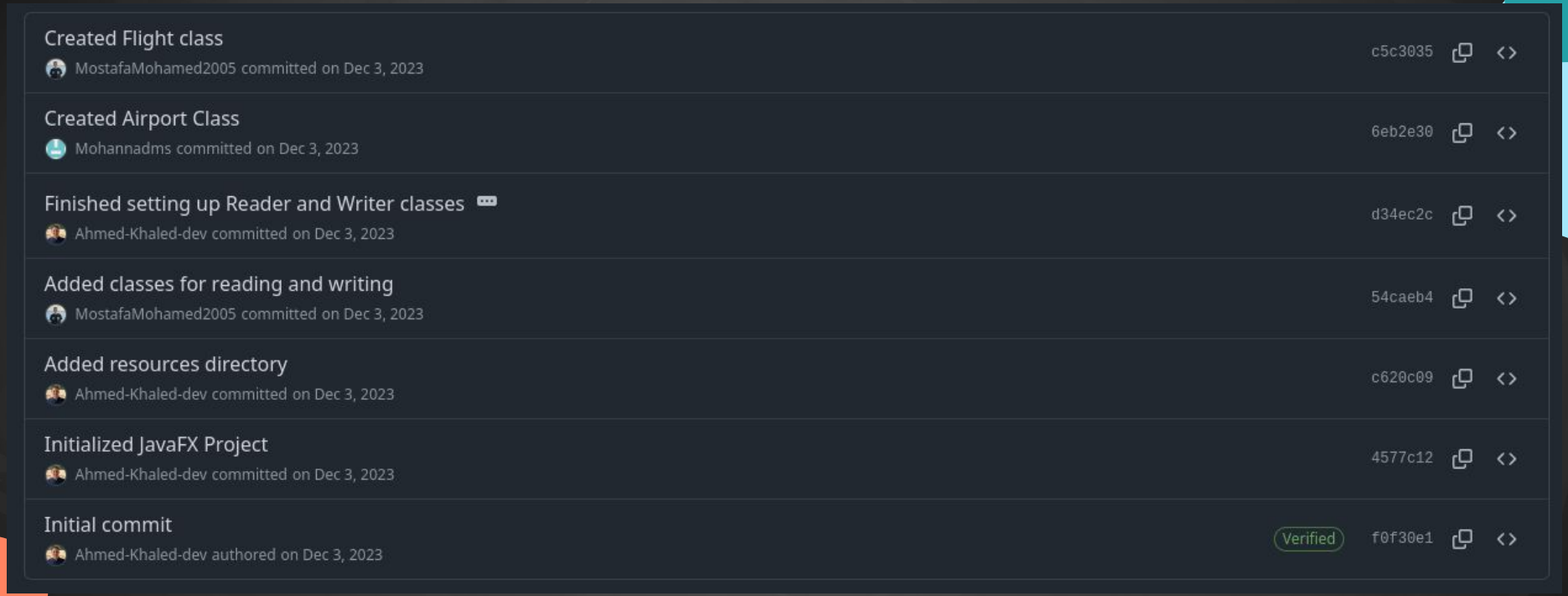
We have UI .... at least some UI
WisdomCasual committed on Dec 12, 2023 5f5c235

Commits on Dec 11, 2023

Co-authored-by: CodingPanda166 <CodingPanda166@users.noreply.github.com>
joonyou34 committed on Dec 11, 2023 80337cb

fxfxfxfxfxfxfxfxfxfxfxfxfxfxfx
WisdomCasual committed on Dec 11, 2023 e59dec2


plz work
AymanM04 committed on Dec 4, 2023
67525e1
Trie
CodingPanda166 committed on Dec 4, 2023
83698d1
Co-authored-by: Kareem Hatem <ERRORsOverflow@users.noreply.github.com>
joonyou34 committed on Dec 4, 2023
06d13f0
Merge branch 'master' of https://github.com/KareemEssam7/Facebook-management
iiYusuf committed on Dec 4, 2023
753de3a
block is now blocking
iiYusuf committed on Dec 4, 2023
99aaa4e


Created Flight class
MostafaMohamed2005 committed on Dec 3, 2023
c5c3035
Created Airport Class
Mohannadms committed on Dec 3, 2023
6eb2e30
Finished setting up Reader and Writer classes
Ahmed-Khaled-dev committed on Dec 3, 2023
d34ec2c
Added classes for reading and writing
MostafaMohamed2005 committed on Dec 3, 2023
54caeb4
Added resources directory
Ahmed-Khaled-dev committed on Dec 3, 2023
c620c09
Initialized JavaFX Project
Ahmed-Khaled-dev committed on Dec 3, 2023
4577c12
Initial commit
Ahmed-Khaled-dev authored on Dec 3, 2023
Verified
f0f30e1

Commits on Mar 5, 2024

docs: Added function headers and generated code docs

```
Code documentation was generated using doxygen
```

Ahmed-Khaled-dev committed on Mar 5 · ✓ 3 / 3

b2853f7

feat: Generate short/full .md release notes from merge commits

Ahmed-Khaled-dev committed on Mar 5

bbff2eb

Commits on Mar 4, 2024

refactor: Split the code to smaller functions

Ahmed-Khaled-dev committed on Mar 4

2eb0e7b

Commits on Mar 3, 2024

feat: Generate .md release notes from normal commits based on type

```
The release notes markdown contains sections, each section represents a conventional commit type (Bug fixes (fix), New Features (feat),
Code Refactoring (refactor)) each section contains the commit message of the normal commit without the commit type and with the first
letter of the commit message capitalized
```

Ahmed-Khaled-dev committed on Mar 3

35133ca

Commit Type	Title	Description	Emoji
`feat`	Features	A new feature	✨
`fix`	Bug Fixes	A bug Fix	🐛
`docs`	Documentation	Documentation only changes	📚
`style`	Styles	Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)	💎
`refactor`	Code Refactoring	A code change that neither fixes a bug nor adds a feature	📦
`perf`	Performance Improvements	A code change that improves performance	🚀
`test`	Tests	Adding missing tests or correcting existing tests	🚨

Main Git Commands

- git log
- git checkout
- git restore (--staged)
- git reset (--soft/mixed/hard)

Initialization

First, we need to ask Git to keep an eye on our directory. From within the directory, run:

```
git init
```

This creates a subdirectory name .git that will hold all the files needed by Git.

Staging Changes

```
git add <what to add>
```

- File name: `file.txt`
- Path to directory: `.` (current directory), `mydir/`
- Pattern: `*.cpp`, `*/`
- All files in the directory: `-A` or `--all`

Viewing Changes

```
git status
```

Ignoring Files

What if there are files you don't need to track?

You can write them in a file named `.gitignore`, and git will simply ignore them.

Committing Changes

When you're satisfied with your staged changes, run:

```
git commit -m [commit message]
```

This will save the state of your staging area as a checkpoint in history.

Viewing History

```
git log [--oneline]
```

This shows the commit history, including commit ID, author, timestamp, and message.

With the option `--oneline`, only ID and message are shown per commit, and each commit is displayed on a single line..

Unstaging Changes

If you no longer want a staged change to be committed, run:

```
git restore --staged [what to remove]
```

This will remove all changes from the staging area, but will keep them in your working directory.

- File name: `file.txt`
- Path to directory: `.` (current directory), `mydir/`
- Pattern: `*.cpp`, `*/`

Undoing Commits

```
git reset [--soft] [commit]
```

This resets out timeline to the specific commit.

The option --soft is used by default, and means that the changes after the given commit will be removed from the commit history, but will remain in the staging area and working directory.

Undoing Commits

```
git reset --mixed [commit]
```

This resets out timeline to the specific commit.

Using the option --mixed means that the changes after the given commit will be removed from the commit history and staging area, but will remain in the working directory.

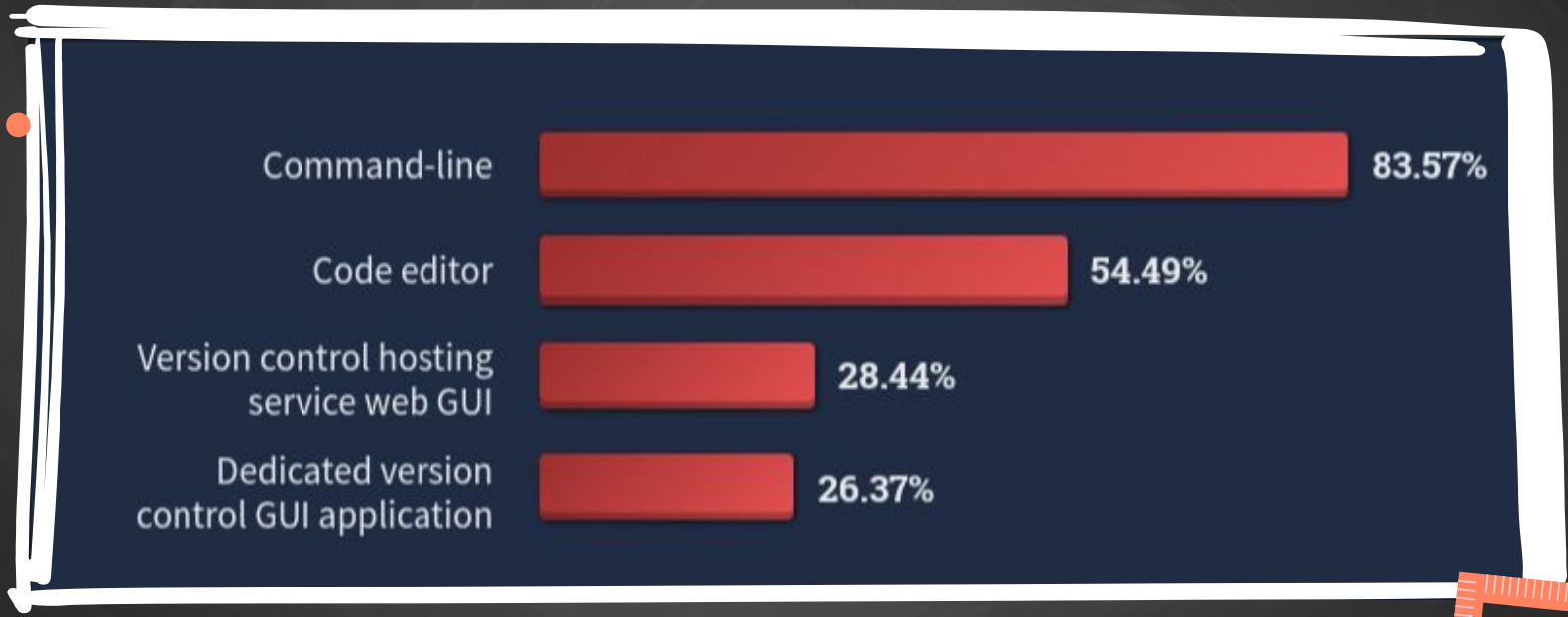
Undoing Commits

```
git reset --hard [commit]
```

This resets out timeline to the specified commit.

Using the option --hard means that the changes after the given commit will be removed from the commit history, staging area and working directory. **Use this with care!**

How Developers Interact With VCSs*



* Stack Overflow 2022 Developer Survey

Hands-On

Hands-On

1. Create a folder named "project", move into it, and initialize it as a Git repository.
2. Create a file called “README.md” and write something in it.
3. Check the Git status and stage the file for commit.
4. Commit the staged file with a message describing the changes.

Hands-On

5. Create another file called “todo.txt”, then stage it and commit it.

6. Edit “README.md” and check the Git status.

7. Restore “README.txt”to the last committed version.

8. You regret adding“todo.txt”. Go back to the first commit.

Agenda

01 Introduction to Open Source ✅

02 Contributing to Open Source ✅

03 What is Git & its Benefits ✅

04 Git Workflow ✅

05 Main Git Commands ✅

Next Session Agenda

01 Branches

02 Conflict Resolution

03 GitHub

04 Managing Repos

05 Forking

06 Pull Requests

Gentle Reminder

Git is a MUST for great software students, not optional.

If you will take one thing from this session, it should be to NEVER work again without using Git.

Hope you enjoyed the session!