

Verzování

SSPŠ
OSE3 Webový backend
2025 / 2026
Ing. Matěj Cajthaml





Verzovací systém

Aneb proč si ukládat verze

Verzovací systém

- Version control
- Správa změn v čase v projektu
- Revize
- Často i ve službách
- Co verzovat
- Proč verzovat
- Primitivní verzovací systémy



Způsob práce

ZAMYKÁNÍ SOUBORŮ

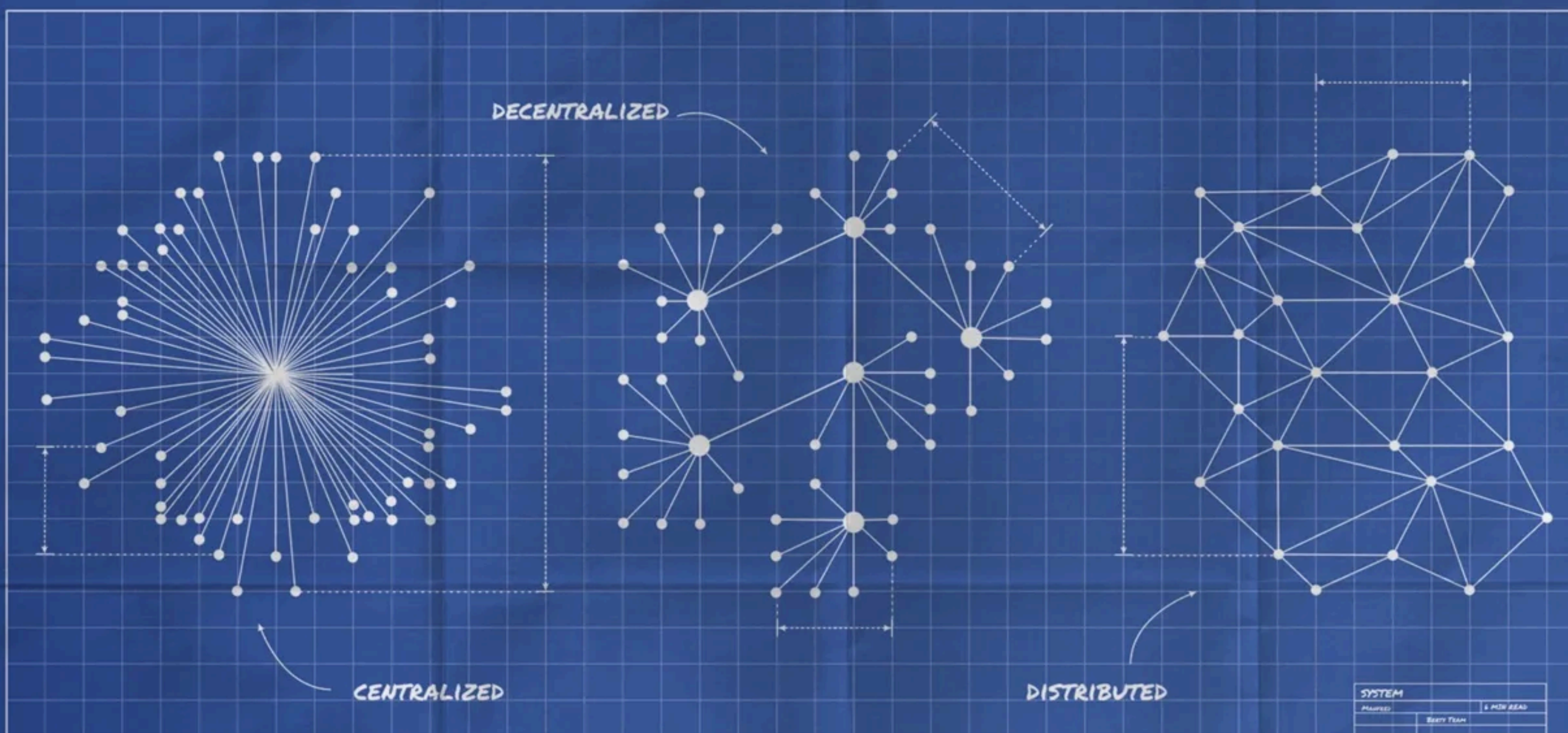
- Celý projekt či soubor
- Pouze jeden člověk pracuje
- Ochrana proti zásahu ostatních

SLUČOVÁNÍ ZMĚN

- Více lidí pracuje najednou
- Změny se spojují
- Problém se spojením
- Vznikají merge conflicty

Umístění

- Centralizované
- Decentralizované
- Distribuované
- Komunikace jen když je potřeba



SYSTEM		
PLAYED:		& POW READ
	BERTY TEAM	

CENTRALIZED VS DECENTRALIZED VS DISTRIBUTED SYSTEM



Známé verzovací systémy

- Git
- Subversion
- CVS
- Bazaar

Typy historie

LINEÁRNÍ

Každá verze závisí na jedné předchozí

GRAFOVÁ

Jedna verze může navazovat na více verzí.
Orientovaný acyklický graf



Git

Aneb jak se z nepříjemnosti stane populární software

Git

- Verzovací systém
- Extrémně populární
- Vzniknul s tím, že staré systémy nebyly dostačující
- Vyžadovaly neustále připojení a celkově byly pomalé
- Vznik open-source řešení
- Linus Torvalds, Junio Hamano, 2005



Git a jeho části

- Každý objekt jasně identifikován SHA-1
- Různé OS a systémy
- Závislost zejména pouze na HW
- Lze používat pouze lokálně
- Ukládání snímků - celé změny
- .git složka
- Working directory
- Staging area
- Instalace Gitu
- CLI vs. GUI



CLI Gitu

- Příkaz git a jeho podpříkazy
- Po instalaci nutné nastavit jméno a e-mail

```
git config --global user.name "Pepa Kopecký"
```

```
git config --global user.email kopecky.p@seznam.cz
```

```
git config --list
```

```
git config user.name
```

```
git help <prikaz>
```



OTÁZKA

Git má cca ~150 příkazů. Musíte si je všechny pamatovat?

Repozitář

- Samotný projekt a jeho všechny verze
- Sleduje veškeré změny (které dovolíme) od počátku sledování
- Nutné vytvořit ručně nebo klonováním
- V aktuální složce pomocí příkazu
- Mohou existovat soubory
- Vytvoří složku `.git`
- Smazáním složky ztratíte vše

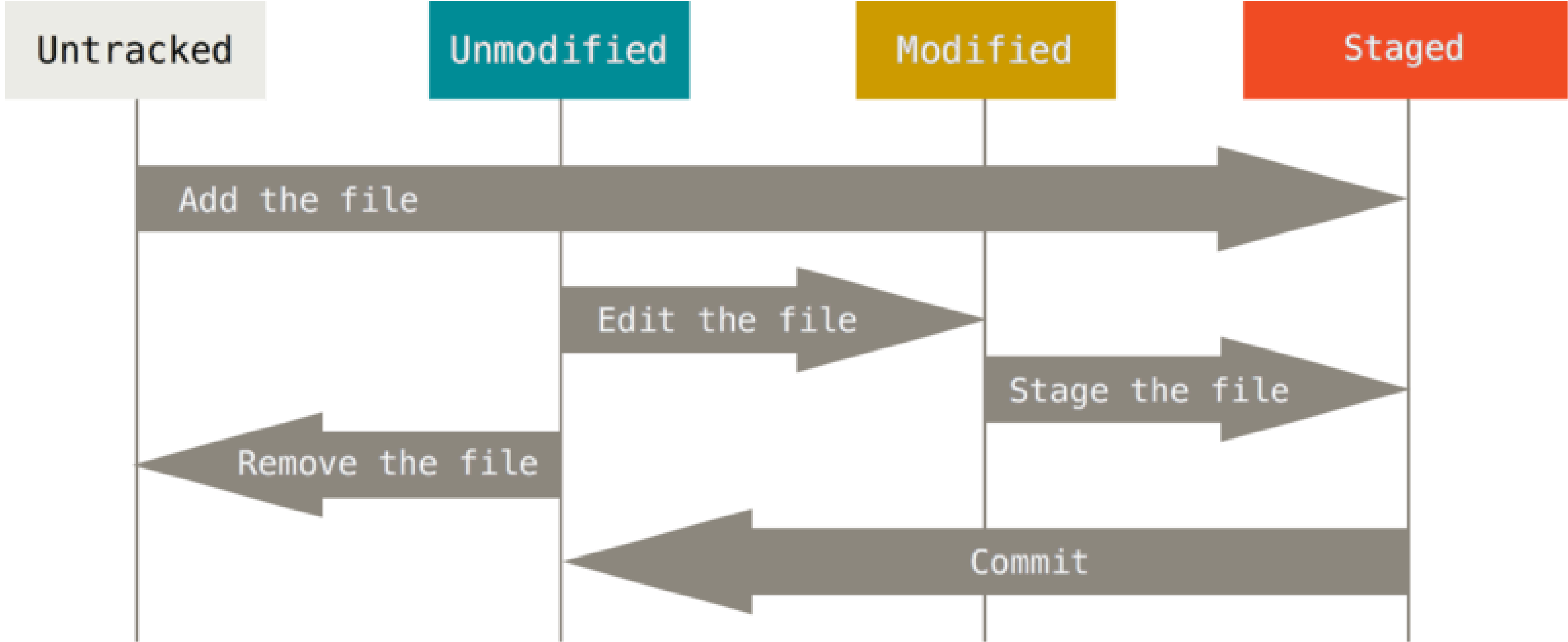
```
git init
```


Commit

- Balíček změn v souborech
- Již zaznamenané a ukončené
- Musí mít jméno, datum vytvoření a autora
- Může mít popis

```
1 git status
```

- **untracked** – nesledován, nový
- **unmodified** – sledován, ale bez změny
- **modified** – sledován, změnil se, ale není označen do commitu
- **staged** – sledován, připraven do commitu
- Zjištění stavu



Změna stavu

```
1 git add <soubor>
2 git add README.md
3 git add path/to/another/file.txt
```

```
1 git restore --staged <soubor>
```

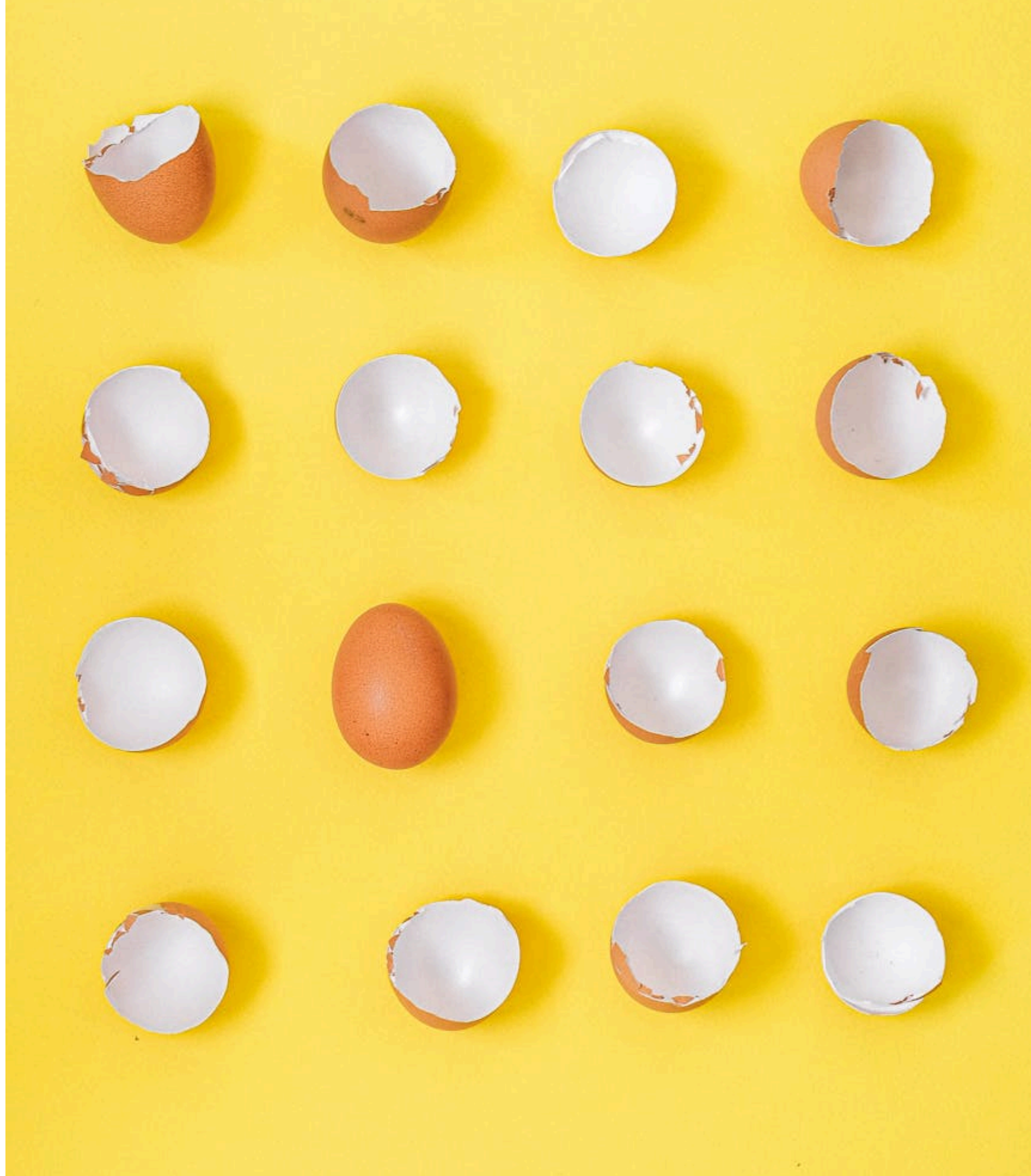
```
1 git add -A
2 git add .
3 git add -u
```

- Přidá do staged
- Označuje se pouze stav v době volání!
- Přemění stav ze staged na modified
- Označí vše
- Označí vše ve složce a rekurzivně
- Označí modifikace a smazání

Kompletní zobrazení změn

- Zobrazí změny dle řádků
- Formát tzv. patch
- Porovná poslední commit s tím, co máme teď ve working directory
- Přepínač --staged
- Porovná staged soubory s posledním commitem

```
1 git diff
2 git diff --staged
```



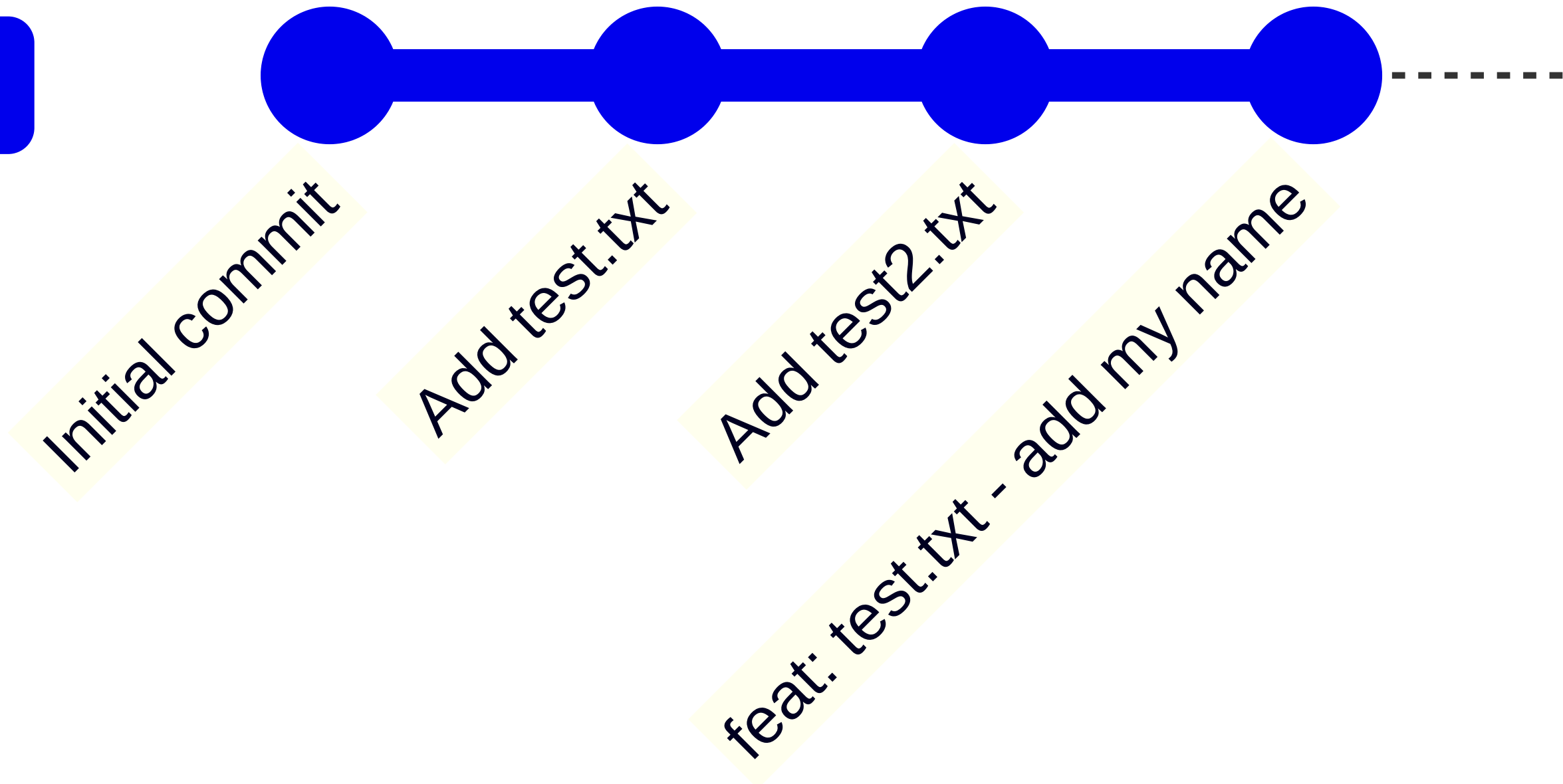
Zapsání změn

- Zapsání staged změn jako jednu revizi
- Bez zprávy se otevře určený editor se souborem – po zavření se provede commit
- Nejlépe udržovat konzistenci
- Po příkazu se vyčistí staged a lze opakovat

```
1 git commit  
2 git commit -m "zprava"
```



main



OTÁZKA

Umí Git zaznamenat přesunutí souboru?
Jak v Gitu smažeme soubor?

Ignorování souborů

- Nějaké soubory nechceme mít ve verzování
- Např. nastavení, configy, knihovny a další
- Speciální soubor .gitignore
- Lze ho mít vícekrát v různých složkách
- Existují předvytvořené od komunity

```
1 # Ignoruj všechny soubory (v jakékoliv
2 *.log
3
4 # Negace pravidla - přestože výše zak
5 !dev.log
6
7 # Ignoruj složku build v hlavním adres
8 /build/
9
10 # Ignoruj soubor build v hlavním adres
11 /build
12
13 # Ignoruj všechny složky lib (v jakéko
14 lib/
15
```

Historie

- Teoreticky dobré ukládat
- Prakticky k verzím potřebujeme přistupovat
- Příkazem historie zjistíme commity, na které budeme referovat v dalších
- Velmi mnoho přepínačů
- Chronologické

```
1 git log
2 git log -10
3 git log --stat
4
5 git log --pretty=oneline
6 git log --graph
7 git log --since=2.weeks
8 git log --before="2022-09-01"
9 git log --after="2022-09-01"
10 git log --author="Matej Cajthaml"
11 git log --grep="redirect"
```

```
* commit b91a02dae68526fe0ab5faad246fc5a26752b4a1
Author: Matěj DenOwq Cajthaml <cajthaml.matej@navarru.cz>
Date: Thu Sep 4 12:38:58 2025 +0200

    fix: Handle populated user object in material owner checks

server/src/materials/materials.controller.ts | 6 +++---
1 file changed, 3 insertions(+), 3 deletions(-)

* commit 674535ee6e0b166d266e5b5d5b1c19b7cbcbbc62
Author: Matěj DenOwq Cajthaml <cajthaml.matej@navarru.cz>
Date: Thu Sep 4 12:28:31 2025 +0200

    fix: Prevent error on undefined API response during export polling

client/src/components/editor/dialogs/Sharing.vue | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

* commit d3924e845b83cf86a63261714198871ed81ae3b2
Author: Matěj DenOwq Cajthaml <cajthaml.matej@navarru.cz>
Date: Thu Sep 4 12:23:22 2025 +0200

    chore: Bumps application versions

client/package.json | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)

* commit 815736f69c4e6cca78e6fbf278b33712b3ff65f2
```

Resetování souboru

- Soubory často potřebujeme vrátit do stavu minulého commitu
- Např. něco zkusíme a už nechceme pracovat na této verzi

```
1 git restore <soubor>
```


Přidání k poslednímu

- V posledním commitu jsme něco zapomněli
- Můžeme do něj přidat další obsah
- Původní commit se přepíše novým
- Možná oprava jména

```
1 git commit --amend
2 git commit --amend -m "Add redirect"
```

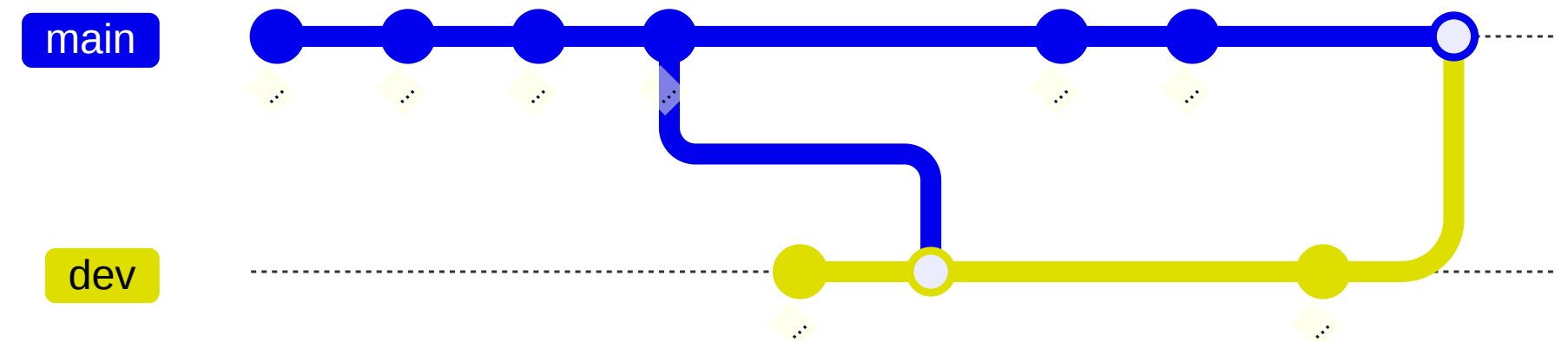


Spolupráce v Gitu

Aneb jak může mít každý svůj klid

Větve

- Oddělení se od konkrétní verze
- V podstatě vlastní linie verzí
- Často slouží k vytvoření změn a pozdějšímu spojení
- Větve se přímo neovlivňují
- Základní větev: main / master
- => nemá speciální význam!
- Větev je interně ukazatel na konkrétní commit



Značka HEAD

- Odkazuje na aktuální větev
- Větev ukazuje na commit
- Zjištění kde aktuálně jsme
- K vidění v `git log`



Větve

- Prozatím vše lokálně
- Vytvoří se větev ale nepřepne se do ní
- Možnost smazat, přepínač -d a -D
- Zobrazení

```
1 git branch <jméno>
2 git branch redirect-rework
3
4 git branch -d redirect-rework
5 git branch -D redirect-rework
6
7 git branch
8 git branch -v
```


Přepnutí aktuální větve

- Přepne do nové větve, ve které budeme dělat změny
- Nesmíme mít nic rozpracovaného
- Resetuje soubory do daného posledního commitu v dané větvi

```
1 git checkout <branch>  
2 git checkout dev
```

Zobrazení historie s větvemi

- `--graph`: Vizuálně ukazuje větve
- `--all`: Ukazuje vše, nejenom dosažitelné z HEAD

```
1 git log --graph --all
```

Slučování větví

- Chceme spojit dvě verze
- Sloučíme jednu větev do jiné
- Směr je zadán do aktuální
- Slučování může být jednoduché:
 - => git sám spojí verze
 - => např. posunutím ukazatele
 - => a nebo ne...

```
1 git merge <jméno>  
2 git merge dev
```

Merge conflict

- Konflikt při slučování
- Často změna ve stejném souboru na stejném místě
- Nějakým způsobem se musí vyřešit pro sloučení
- Git editne soubory se speciální syntaxí
- Nutné k vyřešení v `git status`
- Po vyřešení vytvoření commitu




```
1 <<<<<< HEAD:index.html
2 <div id="footer">
3     <p>Bc. Matěj Cajthaml &copy; 2024</p>
4 </div>
5 =====
6 <footer>
7     <p>Ing. Matěj Cajthaml &copy; 2025</p>
8 </footer>
9 >>>>>> dev:index.html
```

A person is shown from the side, wearing a headset with a microphone, working on a laptop. The person's hands are on the keyboard. The background is a blurred home office with a desk, a small potted plant, and a water bottle. The image has a warm, orange-toned overlay.

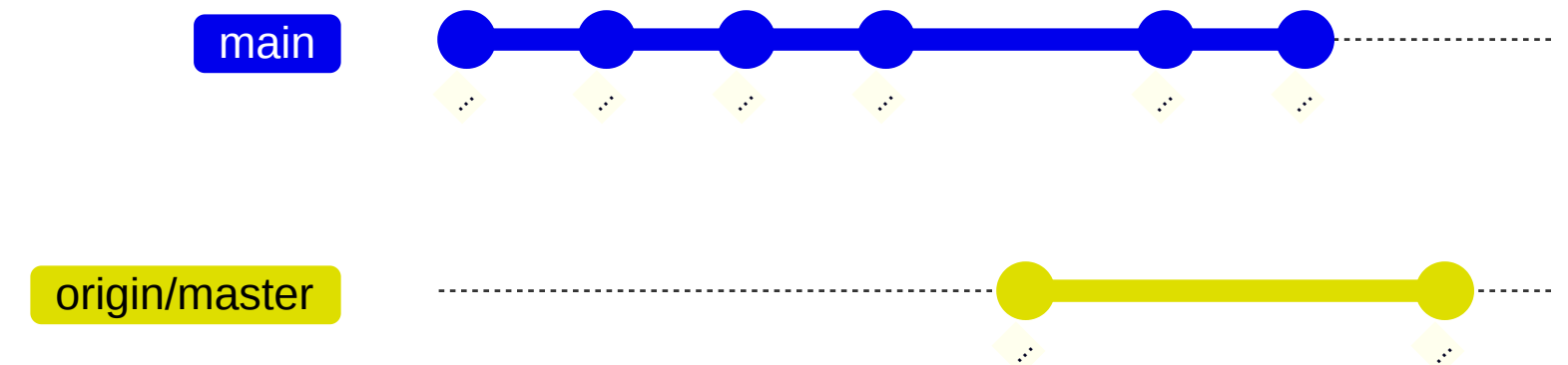
Vzdálená práce

Aneb jak pracovat v různých prostředích

Vzdálené repozitáře

- Git je distribuovaný
- Každý má svoji verzi
- Vzdálené repozitáře dovolují napojení na cizí verze
- Často centralizovaný hlavní repozitář
- Každý má jméno, nejčastěji origin
- Možné je více spravovat, mazat, ...
- Bez konkrétních příkazů vzdálené repozitáře nemodifikujeme

```
1 git remote -v
2 git remote add <nazev> <url>
3 git remote add origin https://github.c
```



Lokální vs. vzdálené větve

- Mohou existovat lokální a vzdálené větve stejného jména
- Např. `main` a `origin/main`
- To neznamená, že jsou propojené!
- Upstream
- Co větev sleduje a kam změny posíláme
- V `status` zjistíme jak na tom oproti vzdálené verzi je

```
1 git branch --set-upstream-to=origin/ma
```


Komunikace se vzdálenými repozitáři

- fetch: Stáhne změny ze vzdáleného repozitáře bez úpravy lokálních verzí
- Tyto změny pak musíme ručně spojit pomocí merge
- pull: Stáhnutí a automatické sloučení

```
1 git fetch <jméno>  
2  
3 git pull
```

Klonování

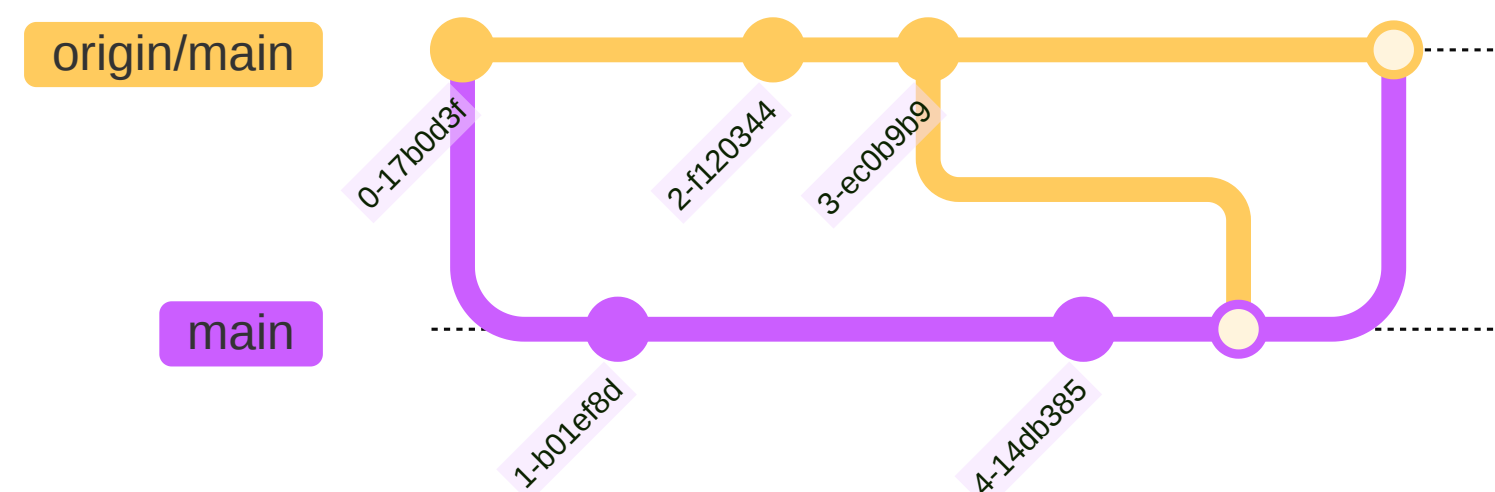
- Stáhnutí vzdáleného repozitáře
- Automatické vytvoření lokálních větví
- A jejich napojení upstreamu
- HTTP/SSH

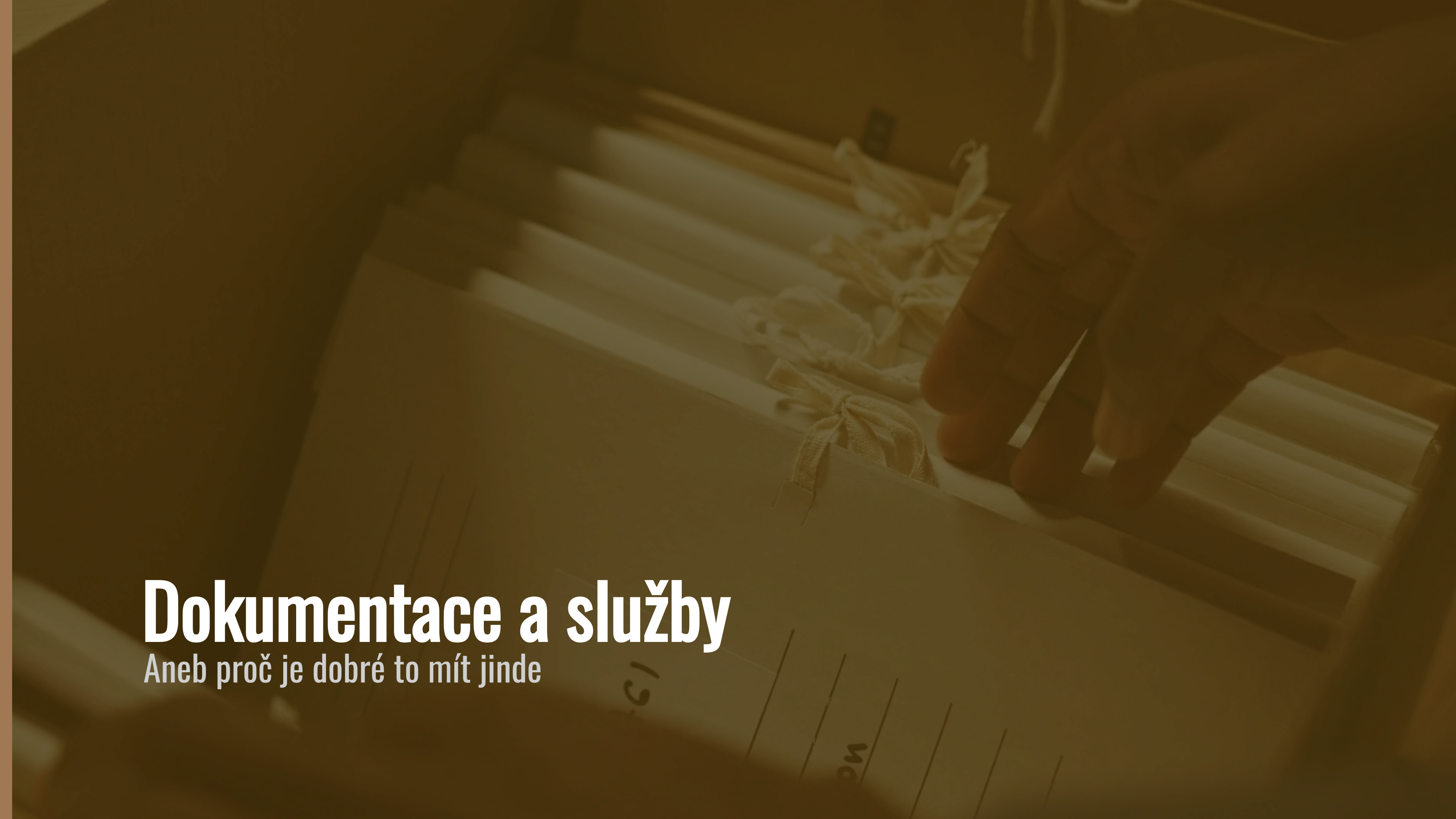
```
1 git clone <url repozitáře>  
2 git clone https://github.com/ssps-cajt
```

Nahrání změn

- Při správném nastavení upstreamu bez přepínačů
- Nutné vytvořit vzdálenou větev, pokud neexistuje
- Vždy stahujeme změny před odesláním

```
1 git push <vzdálený repozitář> <větev k  
2 git push origin master  
3 git push -u origin issue3
```



A hand is shown sorting through a box filled with numerous documents and folders. The entire image is covered with a semi-transparent brown overlay. The documents have some text and lines on them, but they are mostly illegible due to the overlay and focus.

Dokumentace a služby

Aneb proč je dobré to mít jinde

Gitové služby

- Git je verzovací nástroj
- Git ale potřebujeme někde hostovat
- A mít k němu další služby
- Gitová služba = hostování repozitáře a funkcionality na více



OPEN

Gitové služby

GITLAB

- Opensource
- Zdarma, limitované prostředí
- Jednoduchá instalace

GITHUB

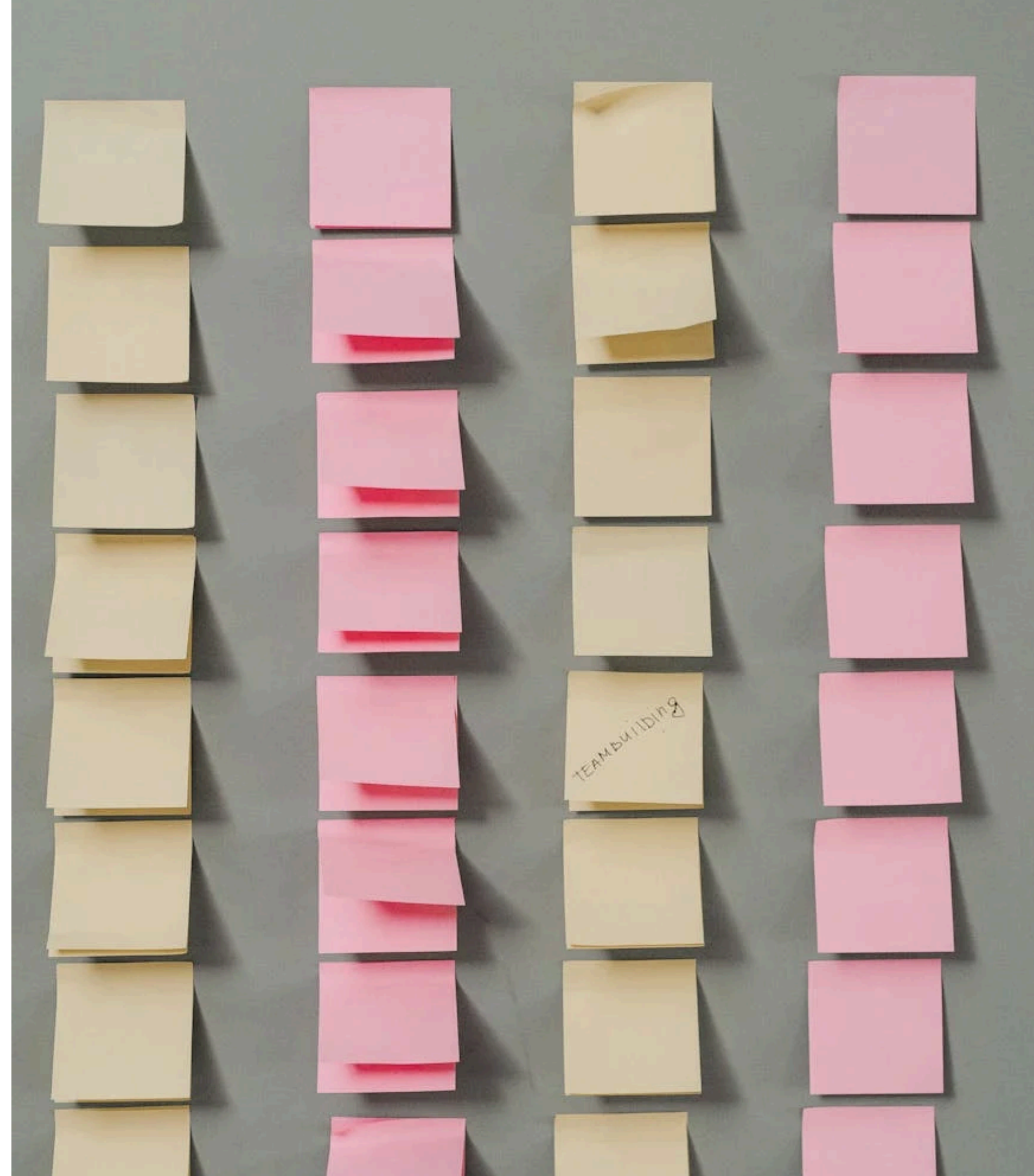
- Zdarma, limitované prostředí
- Nejznámější
- Na ukázkou používáme tuto

Issues

- Možnost zapisovat problémy / buggy / nápady pro projekt
- Možnost diskuze, přiřazení, značek, ...
- Mají vlastní číslo, které můžeme referovat v dalších částech (např. v commitech)
- Vždy konstruktivní a jasné požadavky
- Vlastní šablony

Projects

- Tabulka / Kanban board
- Plán věcí, jejich rozřazení pro lepší organizaci
- Automatizované akce
- Přidáváme issues nebo textové záznamy



Žádosti o spojení

- Merge request / pull request
- Způsob omezení možnosti mergovat
Váš repozitář s hlavním
- Požádáte, aby Vaše větev byla
spojena s nějakou v repozitáři
- Diskuze, různé vlastnosti a možnosti
review
- Jen někteří členové repozitáře
mohou schválit

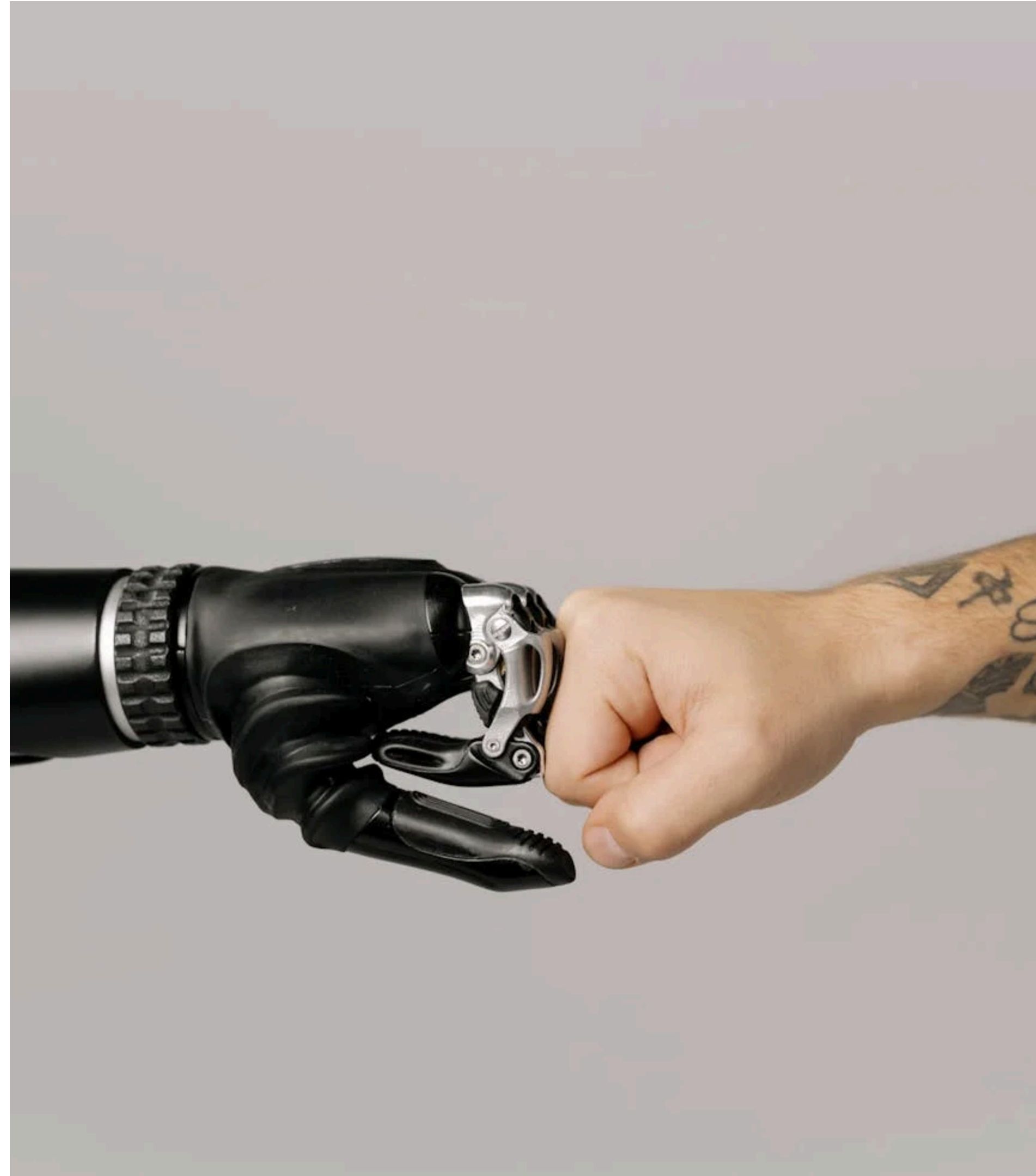


Releases

- Způsob vydání verze
- Automaticky se přikládá kód
- Můžete přiložit jakýkoliv soubor (např. pro stáhnutí)
- Ukazují se na hlavní stránce

Actions

- Spouštění akcí na omezeném prostředí / serveru
- Např. při push se projekt zkompiluje a nahraje na testovací verzi
- Lze spouštět ručně, vidět logy, ...
- Speciální nastavovací soubory
- Spousta komunitně spravovaných rozšíření
- Více o DevOps snad později



OTÁZKA

Jak se v Gitu řeší práva – kdo k čemu má přístup?

Sémantické verzování

- Systém verzování projektů
- Tvořeno ze tří čísel, která se zvětšují
- MAJOR.MINOR.PATCH-metadata
- MAJOR — když nastala změna, která není zpětně kompatibilní
- MINOR — když se přidá funkcionality se zachováním zpětné kompatibility
- PATCH — když se opravila chyba a zůstala zpětná kompatibilita
- Metadata / předběžné verze za verzí pomocí -*+*
- Verze lze řadit

Veřejný repozitář

- Veřejný repozitář implikuje opensource
- To neznamená, že ho můžete upravovat, můžete s ním dělat co chcete a podobně
- Vždy je důležitá licence
- Při přidávání změn je často nutné podepsat smlouvu pro přispěvatele
 - => vzdání se práva na kód,
 - => kód pod danou licencí
- Proces forku

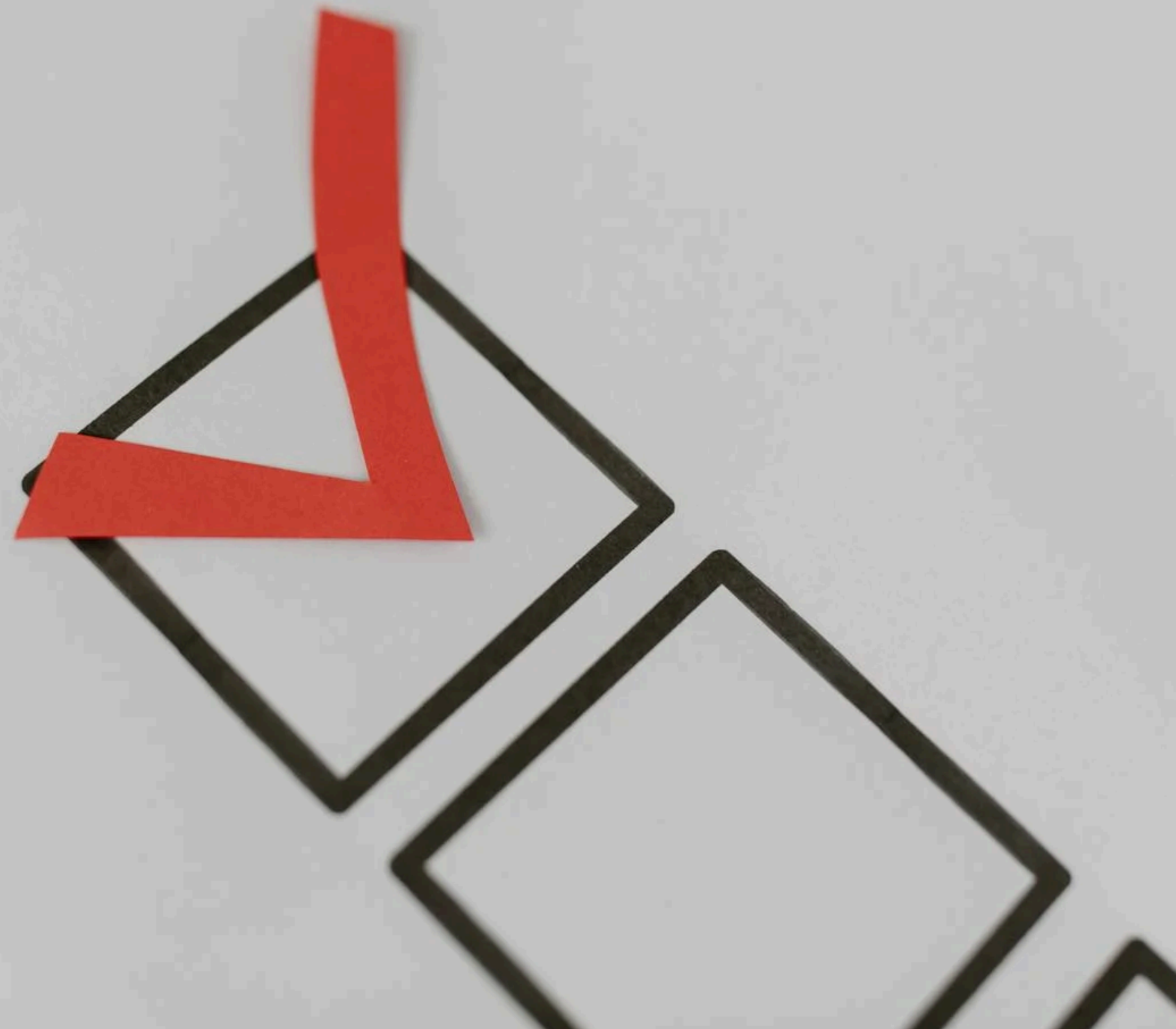


Dokumentace

- Pro projekty je důležité psát dokumentaci
- Jak v kódu, tak i různé návody
- Nějakou dokumentaci lze generovat přímo z kódu (Doxygen, ...)
- Gitové nástroje dovolují pomocí markdownu jednoduché stránky ve Wiki
- README => hlavní soubor, popis projektu, ...

Zásady práce s Gitem

- Každý commit musí být spustitelný
- Zpráva popisuje co commit způsobil
- Každý projekt vlastní způsob práce
- Názvy konzistentní
- Čtení jako kniha
- Větve: trvalé a dočasné
- Issues větve



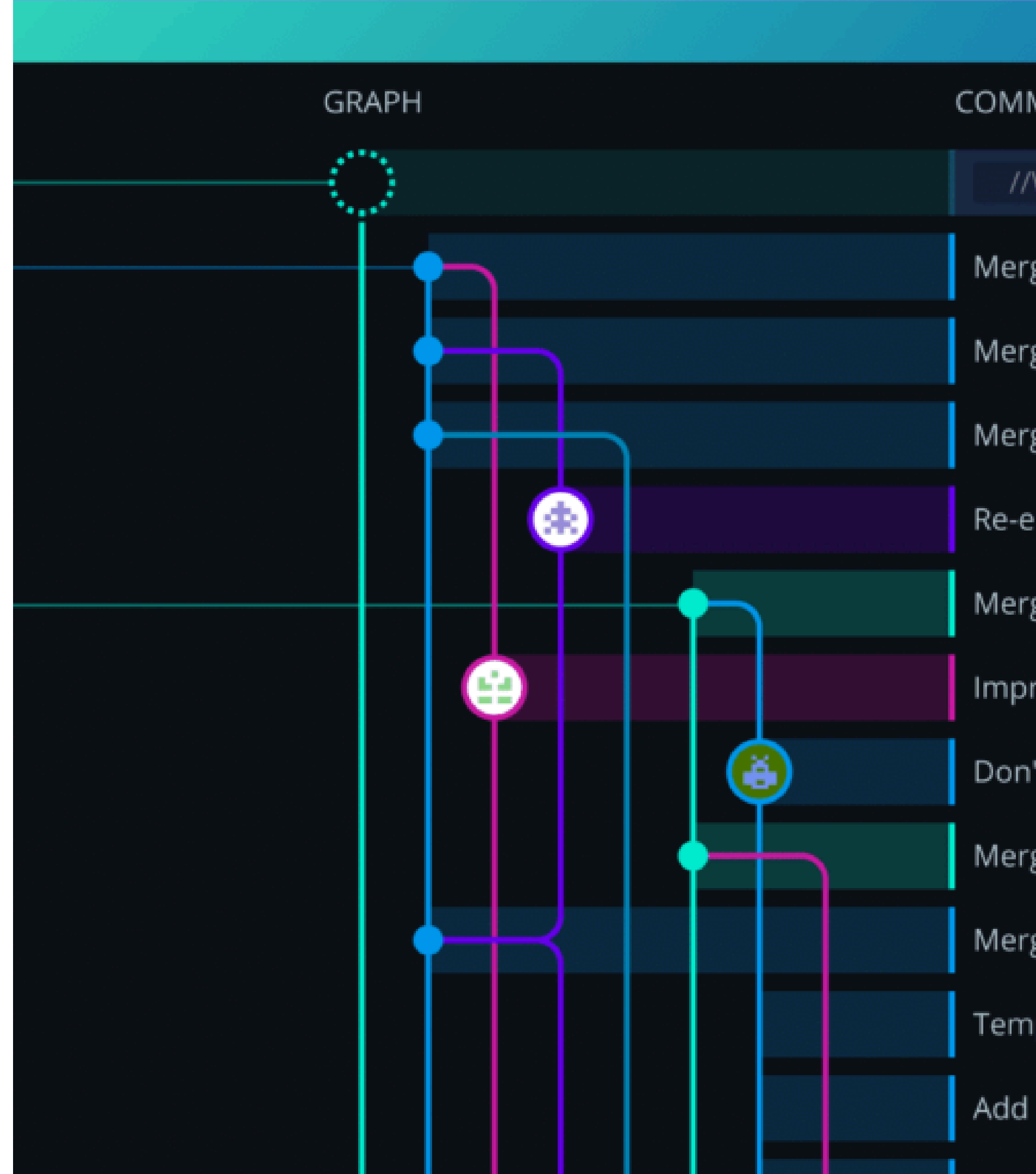


Git klienti

Aneb jak si vše vizualizovat

Git klienti

- Software, program, web. stránka
- Zjednodušená práce s repozitáři
- Není nutné psát příkazy
- Např.: GitKraken, Github Desktop, Sorucetree, obecně IDE



VÝHODY

- Hezké prostředí
- Dobré zobrazení historie
- Intuitivní
- Jednoduché merge conflicty
- Vše to dělá za vás

NEVÝHODY

- Vše to dělá za vás
- Často interakce s něčím, čemu nerozumíte
- Není dobré na pochopení
- Není zdarma, podpora

Co jsme přeskočili

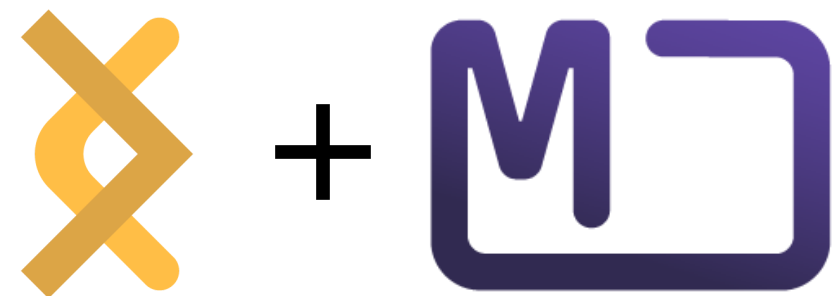
- Další vnitřnosti Gitu a složku `.git`
- Hooks
- Stash, blame
- Přidávání jednotlivých řádek
- Další spojování změn, rebase, cherry-pick, squash
- Resetování a force
- Submoduly



Děkuji za pozornost

matej.cajthaml@ssps.cz

ssps.cajthaml.eu



Vytvořeno pomocí aplikace Materialist

© 2025 Matěj Cajthaml