# Uživatelská konfigurace

▶ Porovnání souborů přes aplikaci Meld

# Úložiště

## Vytvořit na lokálním prostředí

```
git init --bare <cesta>
```

> **Warning**
>
> `<cesta>` = vytvoří úložiště do cesty, musí mít na konci cesty `.git`

## Použít do pracovního prostředí

```
git clone <cesta>
```

> **Warning**
>
> `<cesta>` = adresa k úložišti, musí mít na konci cesty `.git`

> **Tip**
>
> Cesta může být lokální i online.

# Git Submodules

Umožňuje vložit jeden Git repozitář do jiného jako podadresář, přičemž si oba repozitáře zachovávají nezávislost.

Submoduly řeší problém, kdy potřebujete:

- Zahrnout externí kód do svého projektu
- Udržovat přesnou verzi závislostí
- Pracovat na více souvisejících projektech současně

▶ Základní struktura

---

▶ Základní příkazy

---

▶ Praktický příklad použití v Unity projektu

---

▶ Tipy pro práci se submoduly

---

▶ Časté problémy a řešení

---

▶ Výhody a nevýhody

# Git Flow

Git Flow je strategie pro správu větví v Gitu, která usnadňuje práci v týmech a řízení verzí softwaru.

▶ Základní větve v Git Flow

▶ Pomocné větve

▶ Vývoj nové funkce

▶ Příprava vydání

▶ Oprava chyby v produkci

▶ Pravidla pro práci s Git Flow

▶ Vizualizace Git Flow

Pro více informací navštivte oficiální dokumentaci Git Flow

## Aktualizace .gitignore

Odstraňte mezipaměť všech souborů:

```
git rm -r --cached .
```

Jakmile vymažete existující mezipaměť, přidejte/stage soubor/soubory v aktuálním adresáři:

```
git add .
```

Potvrďte změny:

```
git commit -m "Aktualizace .gitignore"
```

## Nová Branch

```
git checkout master       # Přepne se do zdrojové branch
git branch newbranch      # Vytvoří branch a tím se uloží stejné commity z předchozí
aktivní branch
git checkout master       # Přepne se do zdrojové branch
git reset --hard HEAD~3   # Odstraní 3 commity zpět.
git checkout newbranch    # Přepne se do cílové branch
```

Více info zde.

## Existující Branch

```
git checkout existingbranch # Přepne se do cílové existující branch
git merge branchToMoveCommitFrom # Přesune commity ze zdrojové branch
git checkout branchToMoveCommitFrom # Přepne se do zdrojové branch
git reset --hard HEAD~3 # Odstraní 3 commity zpět
git checkout existingbranch # Přepne se do cílové existující branch
```

Více info zde.

### `fixup!`

= Nepoužije zprávu z commitu do opravy

```
git commit --fixup <hashId>
```

nebo

```
git commit -m "fixup! <hashId> notUsedMessage"
```

### `squash!`

= Sloučí zprávu z commitu do opravy

```
git commit -m "squash! <hashId> optionalCustomMessage"
```

▶ Příklad

## Nahrazení Vzdálené Branch z Lokální Branch

1. Přepnout se na novou branch

```
git checkout --orphan latest_branch
```

> **Note**
>
> `--orphan` znamená, že vytvoří branch bez historie commitů

2. Přidat všechny soubory.

```
git add -A
```

3. Provedení commitu.

```
git commit -am "Initialize commit"
```

> **Tip**
>
> `-am` je zkrácený zápis
>
> Je to stejné jako zápis: `--all` `--message "commit message"`

4. Smazat hlavní branch.

> **Warning**
>
> Zjistěte název hlavní větve. (Většinou se jmenuje `master` nebo `main`)

```
git branch -D main
```

5. Přejmenovat aktivní branch na branch z předchozího kroku.

> **Warning**
>
> Zjistěte název hlavní větve. (Většinou se jmenuje `master` nebo `main`)

```
git branch -m main
```

6. Odeslat změny z pracovního adresáře do centrálního úložiště

```
git push -f origin main
```

> **Tip**
>
> `-f` (force) = Historie commitů v centrálním úložišti je nahrazena historií z pracovního adresáře