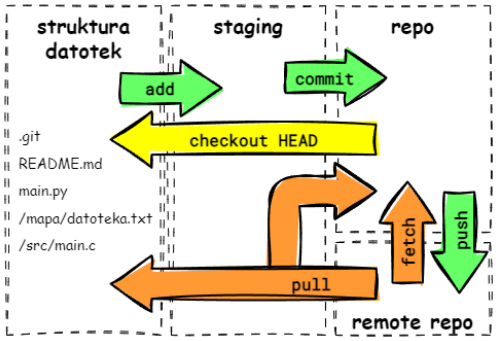
git

Osnovno

Praviloma ima git repozitorij ali repo tri nivoje beleženja sprememb:

- struktura datotek datoteke v mapah, ki so shranjene na disku
- staging vmesna stopnja kamor dodamo datoteke, ki jih želimo dodati v enem commit-u
- repo končna stopnja, kjer so beležene vse spremembe od nastanka repota

Nivoji beleženja sprememb in nekateri ukazi za prehajanje med njimi:



Prenašanje repozitorija

Kadar želimo s strežnika prenesti nov repo:

```
git clone <url/repozitorija>
git clone https://github.com/jankoslavic/pypinm.git
```

SSH

Protokol SSH omogoča varen dostop do oddaljenih repozitorijev bre uporabniškega imena in gesla.
SSH ne potrebujemo, kadar želimo prenesti javni repo.

SSH ključ je sestavljen iz dveh delov:

- **privatni** [`id_ed25519`] predstavlja tvojo podpis
- **javni** [`id_ed25519.pub`] omogoča preverjanje, ali je bil nek dokument podpisan s tvojim privatnim ključem

POZOR! Nikoli ne nalagaj ali pošiljaj svojega privatnega ključa (datoteka brez končnice).

Najprej generiramo nov par ED25519 ključev:

```
ssh-keygen -t ed25519 -C "<komentar>"
```

Na nadaljnja vprašanja lahko odgovorimo z privzetimi vrednostmi.
Javni SSH ključ kopiramo z naslednjim ukazom:

```
cat ~/.ssh/id_ed25519.pub | clip          # Windows
xclip -sel clip < ~/.ssh/id_ed25519.pub   # Linux
```

SSH ključ nato vnesemo na spletni strani GitHub.com. [`Settings -> SSH keys -> Add new key`]

Globalne nastavitve

Nastavi e-mail in Uporabniško ime uporabnika:

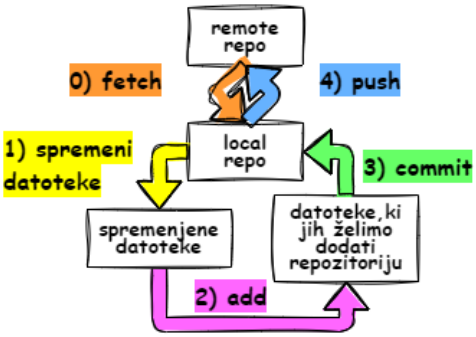
```
git config --global user.name "<uporabnisko_ime>"
git config --global user.email "<tvoj@email.com>"
```

Ustvarjanje novega repoziritorija

```
git init

git remote add <remote_name> <remote_repo_url>
git remote add origin git@github.com:User/repo.git
```

Workflow



Dodajanje sprememb

Ko smo opravili spremembe datotek jih pričnemo pripravljati v staging area.
Dodamo lahko: datoteke poimensko, celotno mapo vse spremembe:

```
git add <ime/datoteke>
git add src/main.py        # Ena datoteka
git add src                # Celotna mapa
git add *                  # Vse spremembe
```

Predhodno izbrisane datoteke lahko dodamo z ukazom `add`. Z ukazom `rm` datoteke hkrati izbrišemo in dodamo spremembe v staging:

```
git rm <ime_datoteke>
git rm src/main.py
git rm -r <ime/mape>
git rm -r src
```

POZOR! Ukaz `git rm` bo datoteko izbrisal tudi z diska.

Shranjevanje sprememb

Ko smo v staging dodali vse spremembe, ki spadajo v neko zaključeno enoto spremembe zabeležimo v lokalni repozitorij:

```
git commit -m "<komentar>"
git commit -m "Dodana funkcionalnost XZY"
```

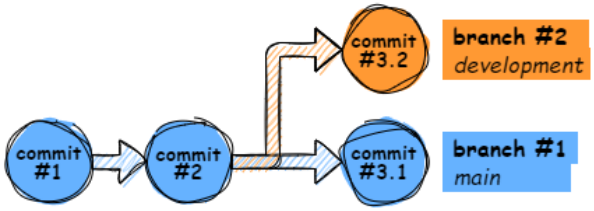
Pošiljanje sprememb na strežnik

Spremembe zabeležene v lokalnem repozitoriju posodobimo tudi na srežniku:

```
git push origin <ime-veje>
git push origin main
```

Veje

Veje omogočajo ločeno okolje za urejanje kode. Primer dobre prakse je, da novih funkcionalnosti ne dodajamo neposredno v glavno vejo (običajno: `main`). V ta namen dodamo novo vejo (npr.: `nova-funkcionalnost`) in novo funkcionalnost razvijamo v tej veji. Ko je nova funkcionalnost končana in testirana jo lahko združimo nazaj v glavno vejo.
Primer nove veje:



Aktivno vejo izberemo z:

```
git checkout <ime-veje>
git checkout main
```

Novo vejo ustvarimo z:

```
git branch <ime_veje>
git branch development
```

Vejo ustvarimo in aktiviramo
(ekvivalentno `brench` + `checkout`):

```
git checkout -b <ime-veje>
git checkout -b development
```

Vejo izbrišemo z:

```
git push -d <remote_name> <ime_veje>
git branch -d <ime_veje>
```