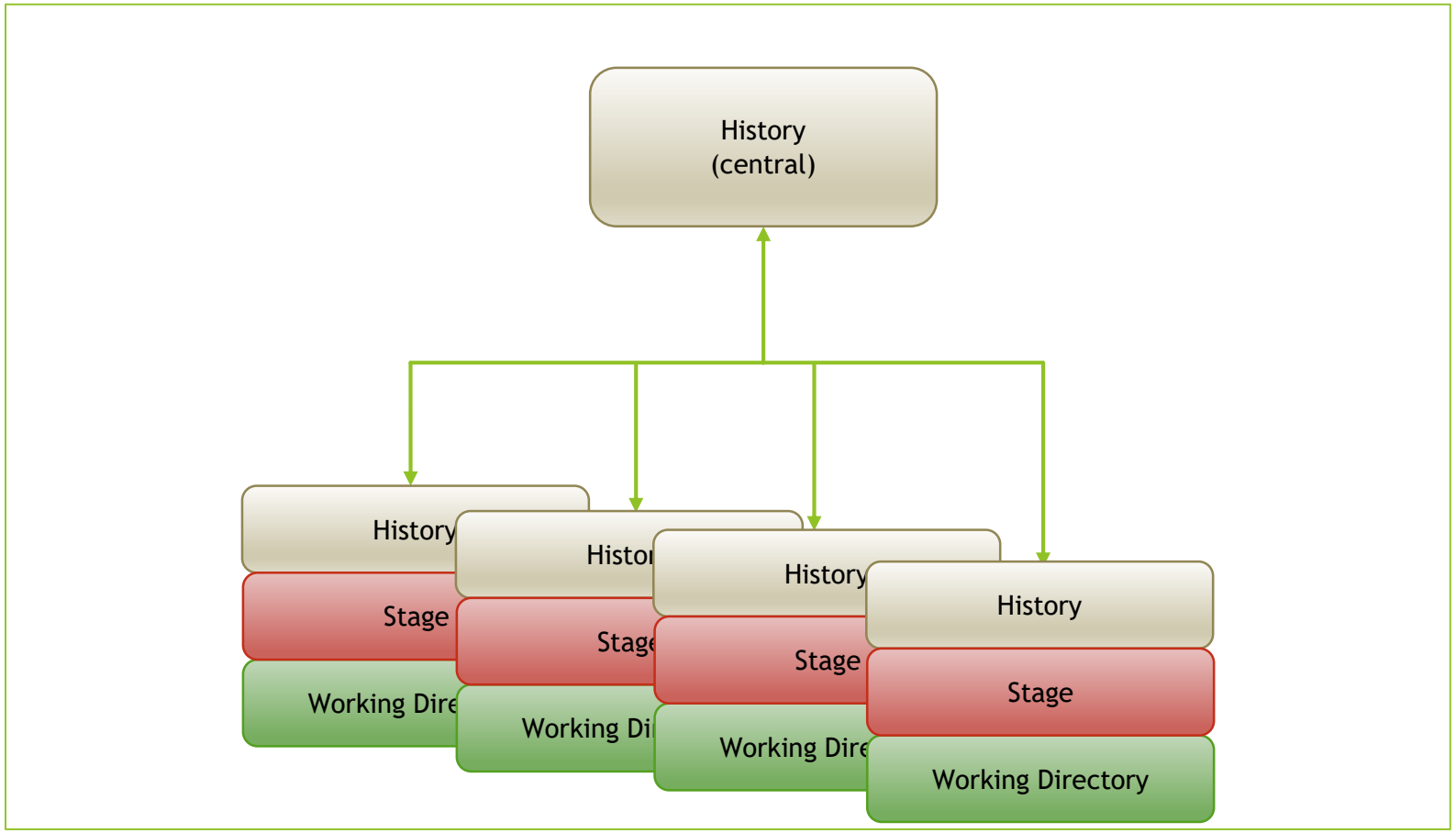
Git

Rozproszony system kontroli wersji

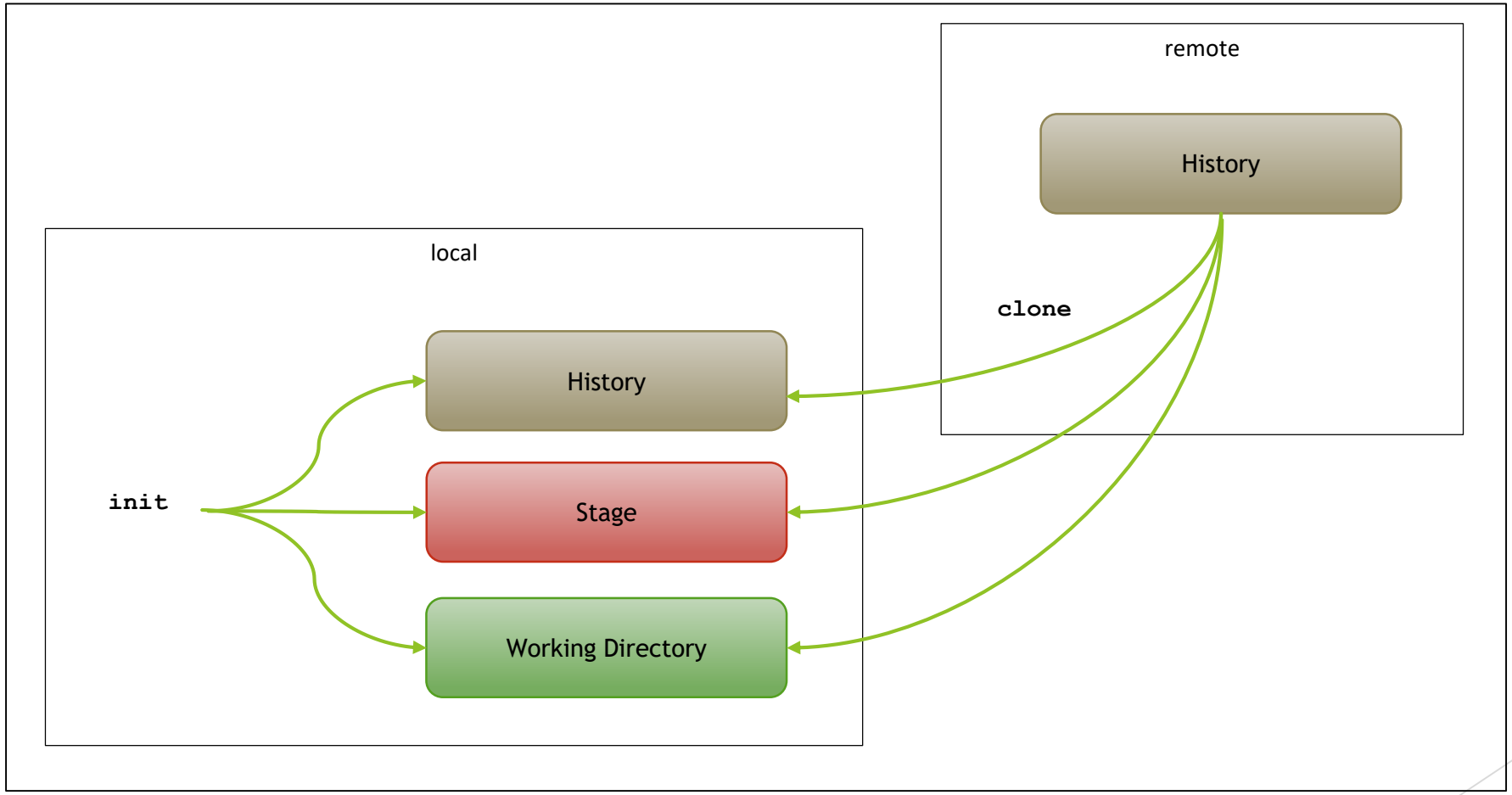
Wstęp

- **Git** jest rozproszonym (DCVS), międzyplatformowym systemem kontroli wersji napisanym głównie w języku programowania Python. Głównymi założeniami Gita były: duża wydajność, skalowalność oraz zaawansowane możliwości operacji na gałęziach. Twórcami są Junio Hamano oraz Linus Torvalds. Kod źródłowy wydany jest na licencji GNU General Public License 2.

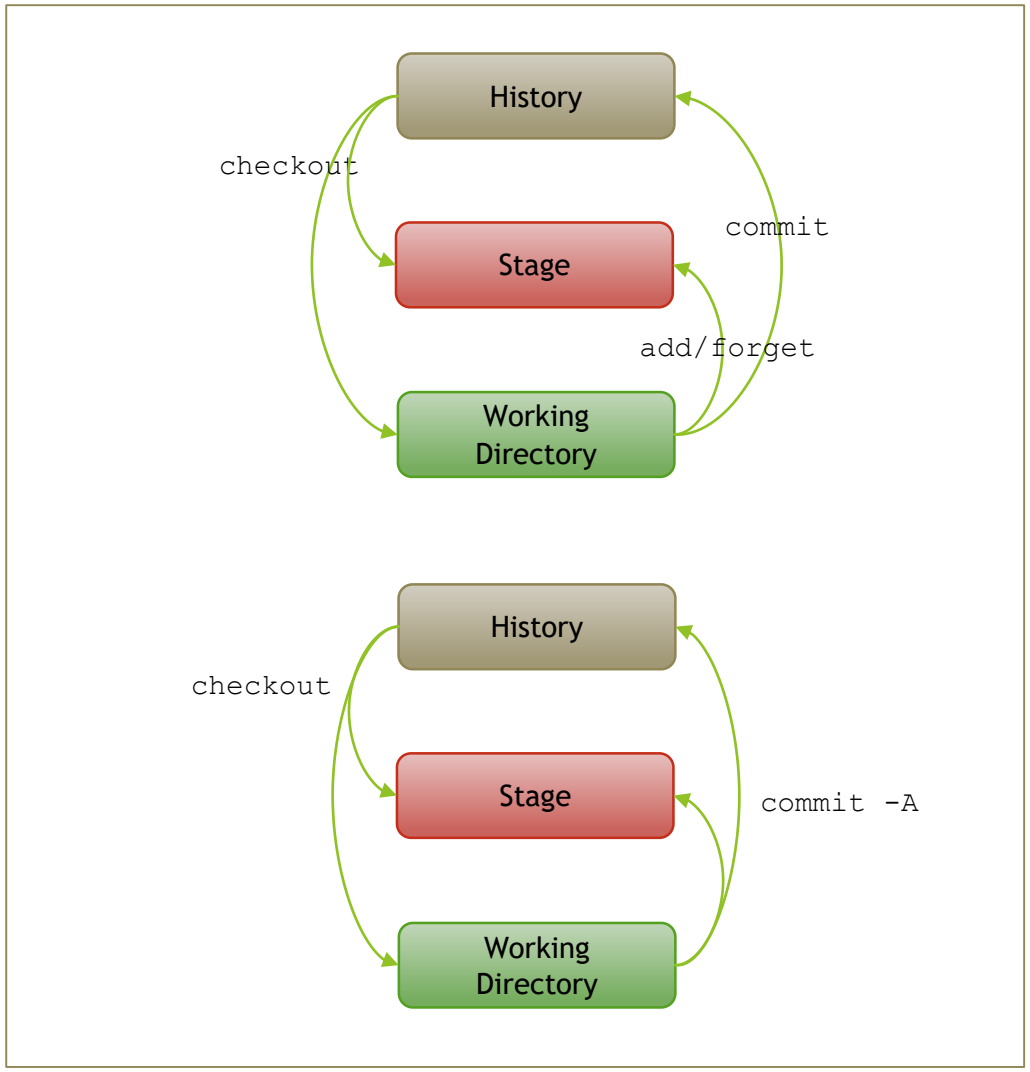
Organizacja repozytorium



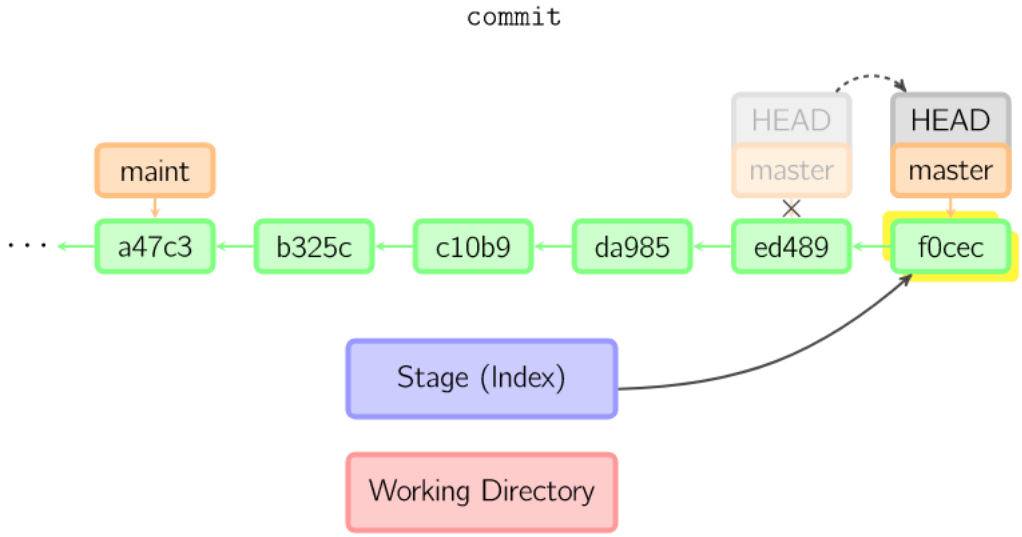
Inicjowanie repozytorium

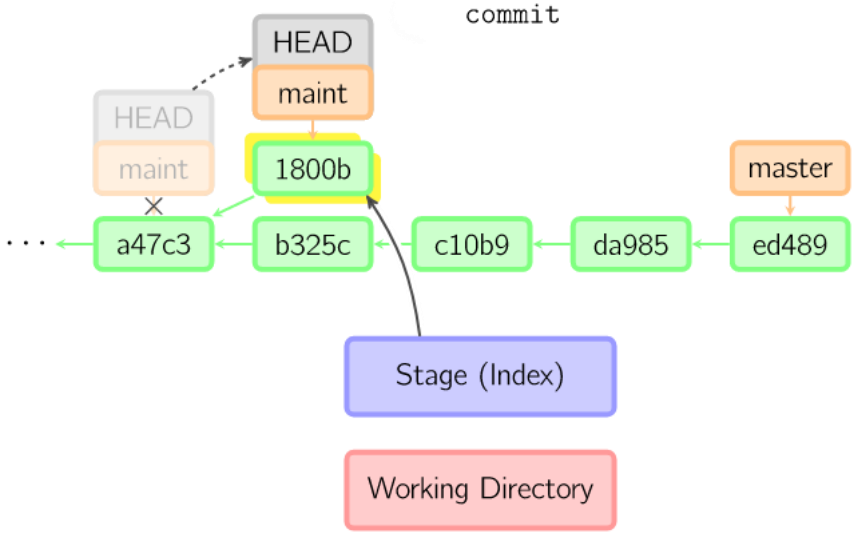


Podstawowe operacje

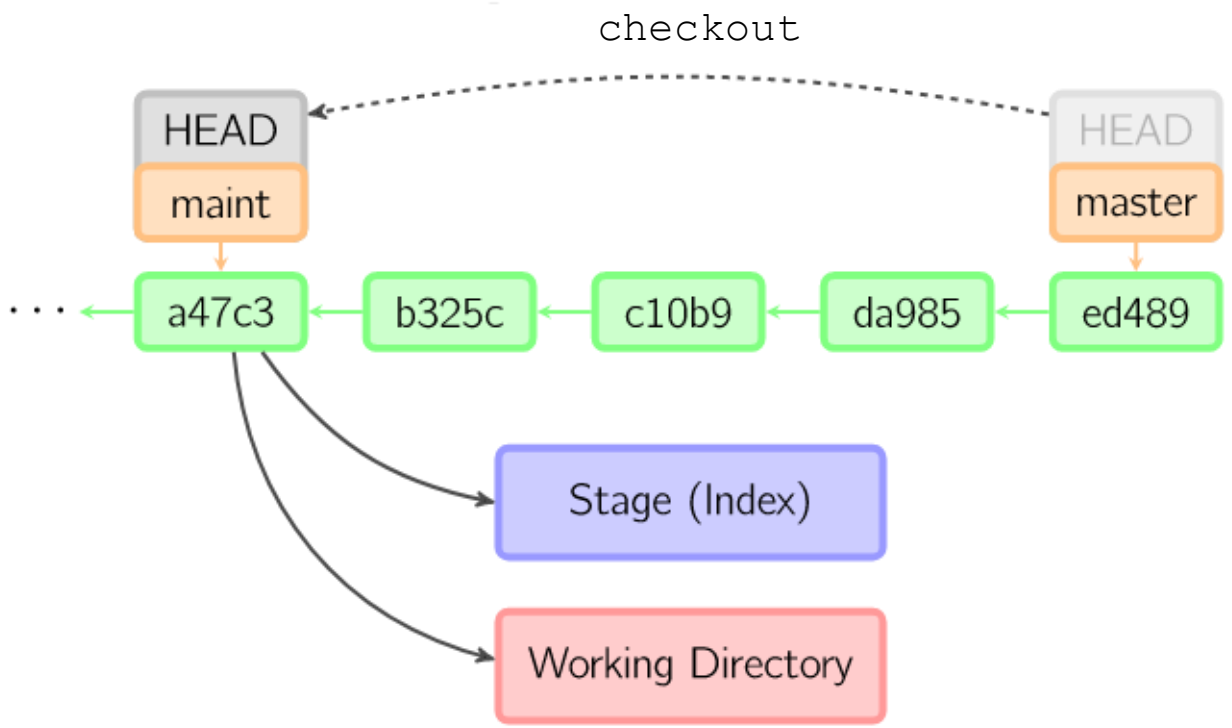


Zapisywanie zmian (commit)



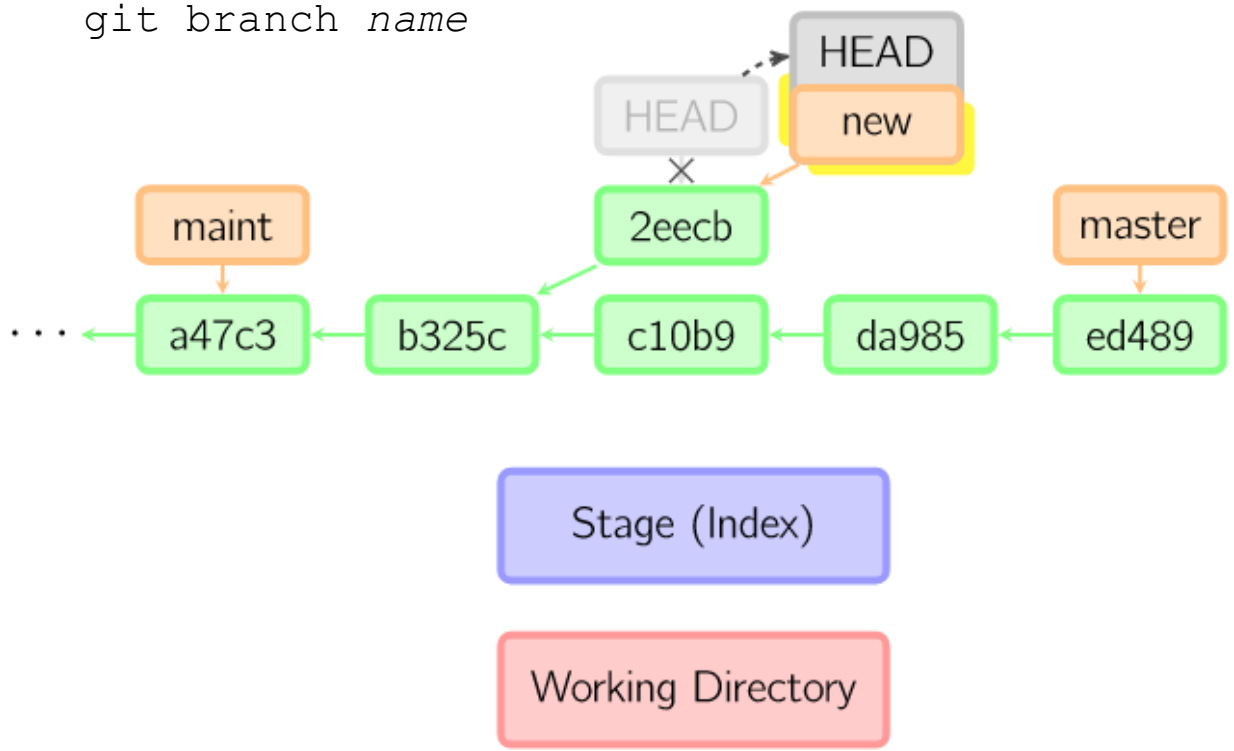


Pobranie zapisanej wersji

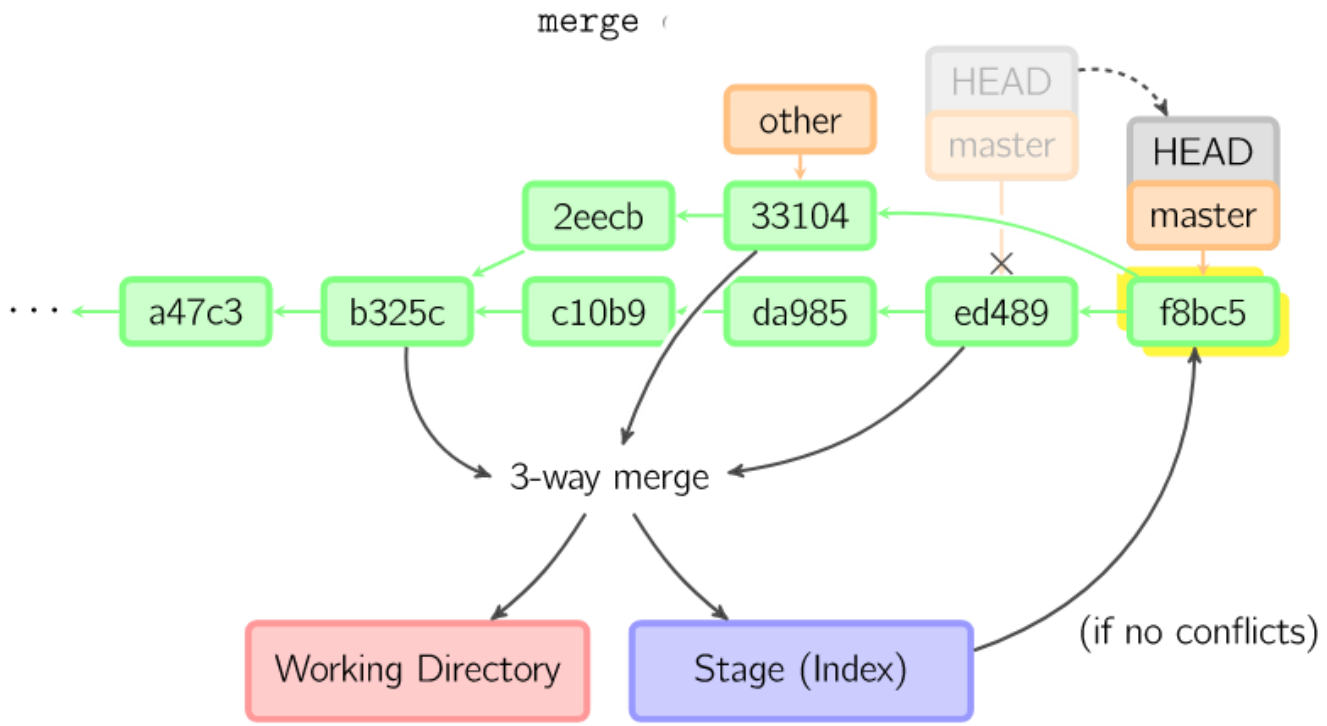


Tworzenie nazwanych gałęzi (branch)

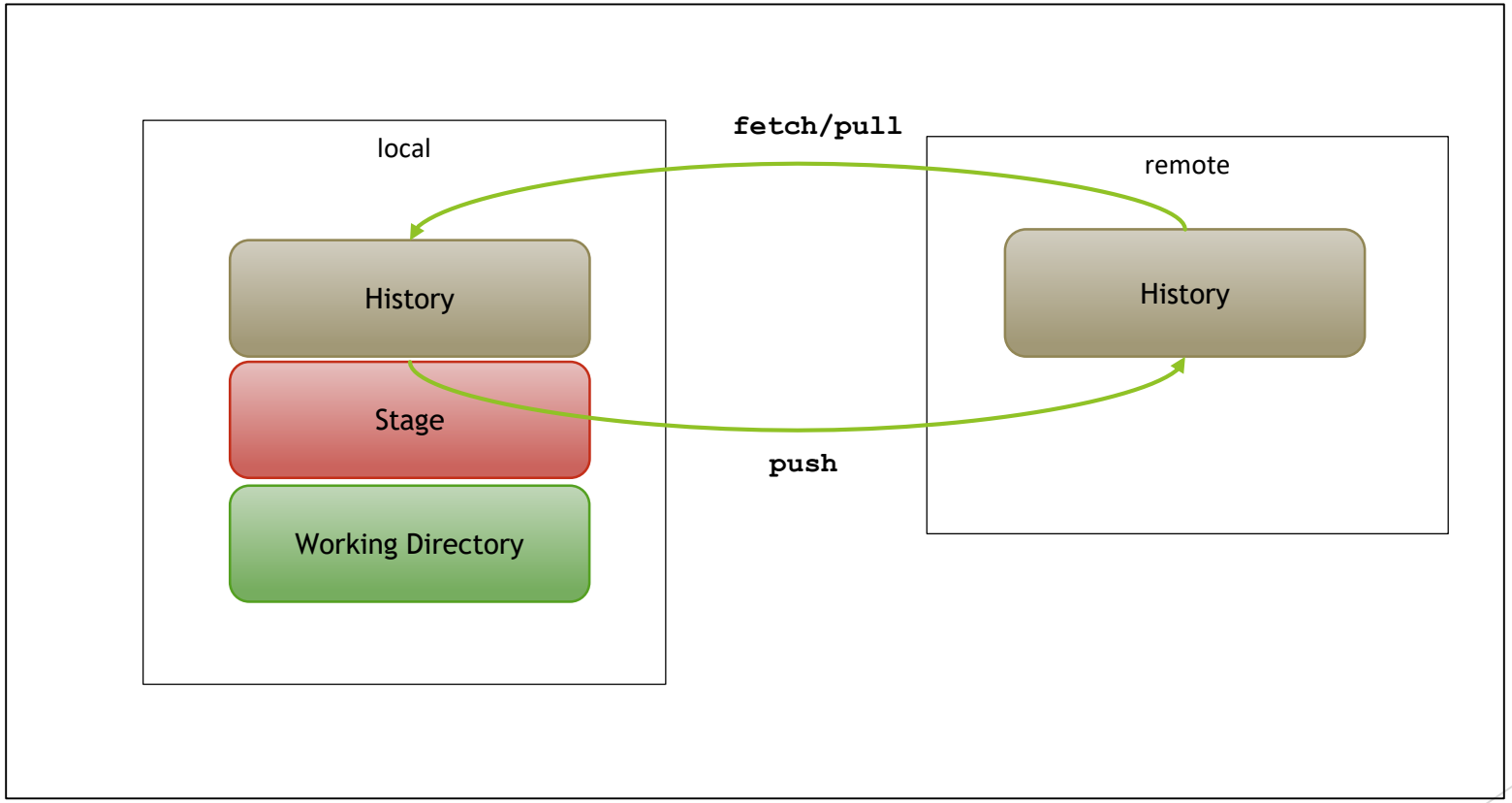
`git branch` *name*



Scalanie zmian (merge)



Synchronizacja repozytoriów (push/pull)



Instalacja git i przydatnych narzędzi:

- Instalacja git:

```
sudo apt-get install git
```

- Instalacja narzędzia porównywania:

```
sudo apt-get install kdiff3
```

- Instalacja git-flow

```
sudo apt-get install git-flow
```

- Opcjonalnie można zainstalować jakiś git-gui np. „smartgit”

Konfiguracja git:

- Definiowanie użytkownika:

```
git config --global user.name LOGIN
git config --global user.email EMAIL
```

- domyślnego edytora i narzędzia porównywania:

```
git config --global core.editor nano
git config --global core.difftool kdiff3
```

- Teraz git jest gotowy do pracy.

Prosty schemat pracy:

- Gałąź „master” – główna gałąź „produkcyjna”, tutaj znajduje się zawsze najnowsza **działająca** wersja oprogramowania.
  Nie wprowadzamy zmian bezpośrednio na tej gałęzi!
- Gałąź „develop” – główna gałąź rozwojowa. Stanowi punkt wyjścia do wprowadzania zmian. Zmiany wprowadzamy w podgałęziach.
- Gałęzie „F*#XX_LOGIN*” – gałęzie utworzone z gałęzi „develop” do wprowadzania zmiany (feature) o numerze „#XX” przez osobę „LOGIN”.

Prosty schemat pracy:

1. Klonujemy repozytorium lub, jeśli istnieje, synchronizujemy z serwerem. Przechodzimy do katalogu w którym chcemy założyć repozytorium i podajemy polecenie:

   git clone [*repository URL*]

   Przechodzimy do katalogu repozytorium i ustawiamy swoją identyfikację aby zmiany w repozytorium miały autora:

   cd *Repo_dir*

   git config user.name "FIRST_NAME LAST_NAME"

   git config user.email MY_NAME@example.com

   Możemy też ustawić swoją identyfikację globalnie, dla wszystkich repozytoriów na danym komputerze:

   git config --global user.name "FIRST_NAME LAST_NAME"

   git config --global user.email MY_NAME@example.com

Prosty schemat pracy:

2. Gdy mamy skonfigurowane lokalne repozytorium możemy przejść do realizacji zadań. Przechodzimy do gałęzi „develop” i tworzymy swoją gałąź (na podstawie zagadnienia z Redmine).

   ```
   git checkout develop
   git checkout -b F#XX_LOGIN
   ```

3. Teraz pracujemy nad kodem co pewien czas zapisując zmiany.

   ```
   git commit –A –m „komentarz”
   ```

4. Gdy funkcjonalność jest gotowa aktualizujemy „develop” i scalamy nasze zmiany:

   ```
   git checkout develop
   git pull
   git merge --no-ff F#XX_LOGIN
   git push develop
   ```

5. Jeśli wszystko się udało mamy zaktualizowane „develop” lokalne i remote.

Prosty schemat pracy z wykorzystaniem Git Flow:

2. Inicjujemy git flow w repozytorium (jeśli nie zostało wcześniej zainicjowane):

   git flow init

2. Gdy mamy skonfigurowane lokalne repozytorium możemy przejść do realizacji zadań. Tworzymy nowy feature (na podstawie zagadnienia z Redmine).

   git flow feature start F#XX_LOGIN

3. Teraz pracujemy nad kodem co pewien czas zapisując zmiany.

   git commit –A –m „komentarz”

4. Gdy funkcjonalność jest gotowa kończymy feature:

   git flow feature finish F#XX_LOGIN

   git push develop

5. Jeśli wszystko się udało mamy zaktualizowane „develop” lokalne i remote.

Kilka przydatnych poleceń git:

Sprawdzanie stanu repozytorium:

```
git status
```

```
D:\GitRepos\test_mb>git status
On branch develop
Your branch is up to date with 'origin/develop'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   DOC/test1.txt
        modified:   DOC/tmp.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Lista ostatnich operacji:

```
git log
```

KONIEC