Programowanie w języku wysokiego poziomu II

Paweł Gliwny

Co będzemy robić?

- Python: OOP, zaawansowane zagadanienia (np. Dekoratory, iteratory), bazy danych, aplikacje web (Flask), dashboard(streamlit/Gradio), … .
- Git
- Docker

Na zaliczenie

- Zbudowanie własnej aplikacji (np. opartej na Flask z bazą danych).

oraz stworzenie obrazu Docker tej aplikacji.

Git

- Git to rozproszony, otwarto źródłowy system kontroli wersji. Umożliwia śledzenie kodu, scalanie zmian oraz powrót do starszych wersji.
- Pozwala na synchronizację zmian z serwerem zdalnym.
- Git stał się standardem branżowym, ponieważ obsługuje niemal wszystkie środowiska programistyczne, narzędzia linii poleceń i systemy operacyjne.
- Git tutorial

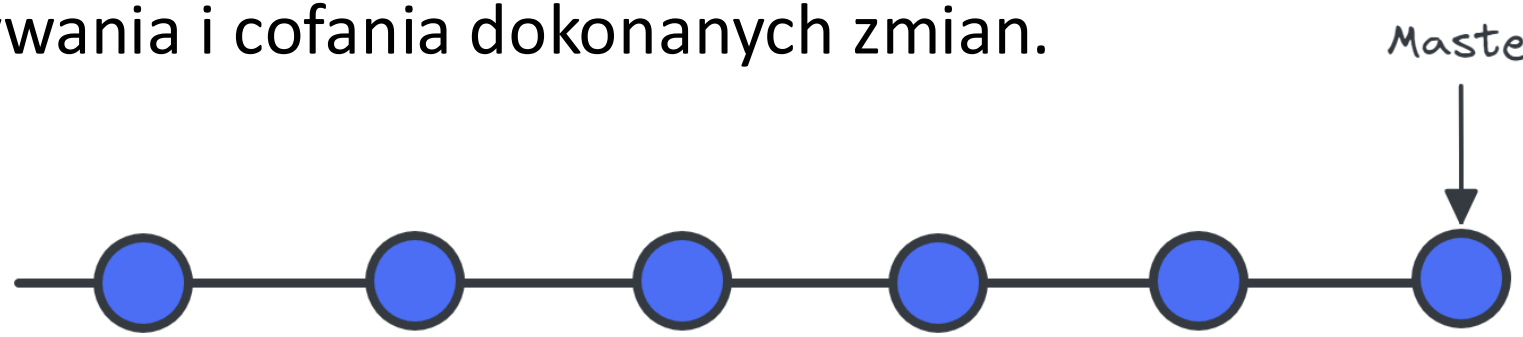
Data Version Control

Zestaw narzędzi i procesów do śledzenia, zarządzania i udostępniania zmian w danych.

- Śledzenie zmian
- Reprodukcyjność
- Skalowalność
- **Popularne narzędzia:**
  - DVC (open-source), LakeFS (chmura), Weights & Biases (platforma)
- **Przykłady zastosowania:**
  - Zestawy danych oraz modele ML
  - Walidacja modeli

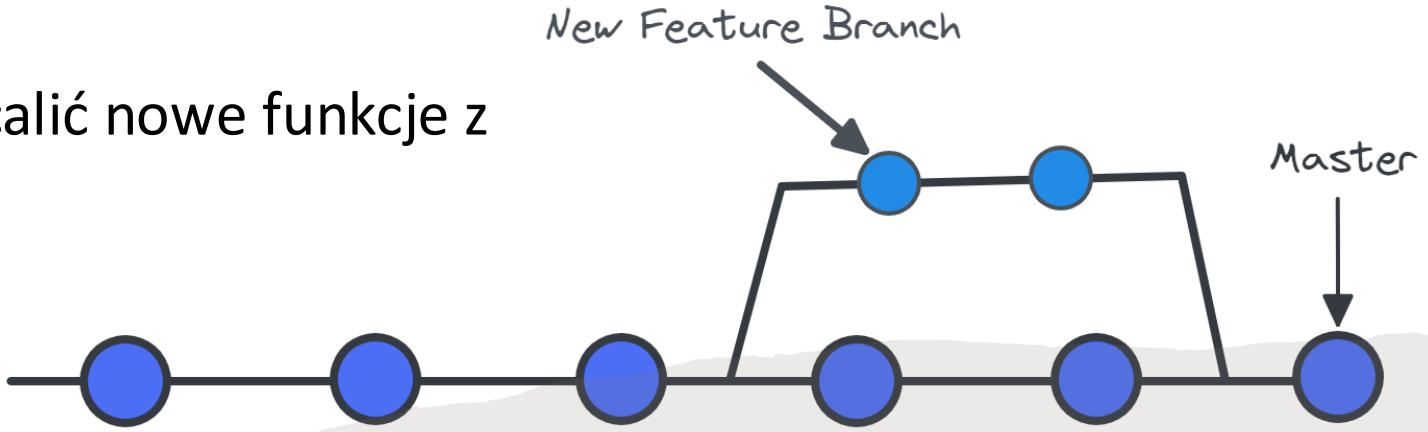
Jak Git działa?

- Git przechowuje pliki oraz ich historię rozwoju w lokalnym repozytorium.
- Przy zapisie zmian, git tworzy **commit** - migawkę plików, łącząc je w graf historii rozwoju.
- Pozwala to wrócić do poprzedniego commita, porównać zmiany i śledzić postęp projektu.
- Commity są identyfikowane przez unikalny hash, który jest używany do porównywania i cofania dokonanych zmian.



Gałęzie

- Są kopiami kodu źródłowego, które działają równolegle do głównej wersji.
- Aby zapisać dokonane zmiany, należy scalić gałąź z główną wersją.
- Ta funkcja promuje pracę zespołową bez konfliktów.
  - Każdy programista ma swoje zadanie i, korzystając z gałęzi, może pracować nad nową funkcją bez ingerencji innych członków zespołu.
- Po zakończeniu zadania możesz scalić nowe funkcje z **główną wersją (gałąź master).**



Commits

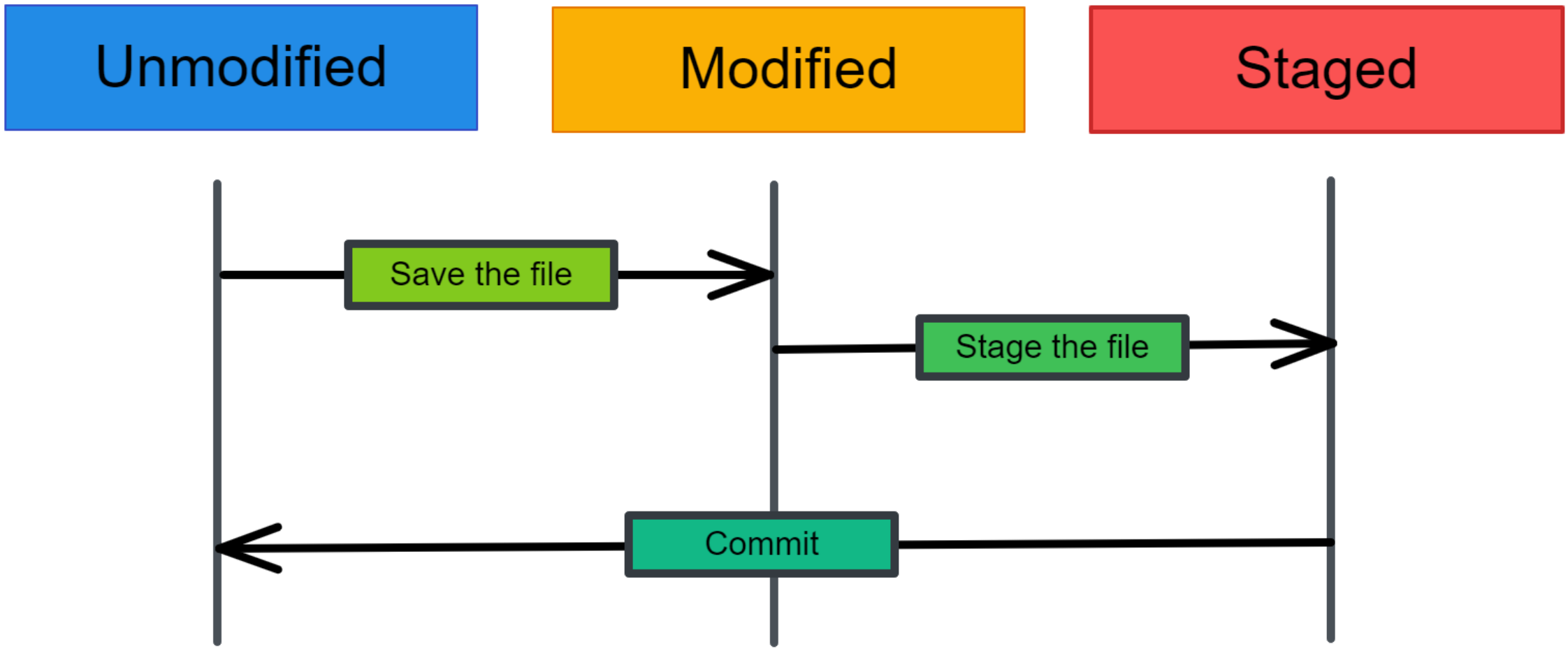
W Git istnieją trzy stany plików:

- zmodyfikowane,
- przygotowane (staged)
- zatwierdzone (commit).

Gdy dokonujesz zmian w pliku, zmiany są zapisywane w lokalnym katalogu. Nie są one częścią historii rozwoju git.

Aby utworzyć **commit**, musisz najpierw przygotować zmienione pliki (***git add***).

  - Możesz dodawać lub usuwać zmiany w obszarze przygotowania, a następnie zapakować te zmiany jako commit z wiadomością opisującą zmiany.


Unmodified
Modified
Staged
Save the file
Stage the file
Commit

Zalety git

- **Śledzenie zmian:** Git umożliwia przeglądanie historycznych zmian w kodzie.
- **Integracja z IDE:** Git jest zintegrowany ze wszystkimi środowiskami programistycznymi, na przykład *VSCode i JupyterLab.*
- **Współpraca w zespole:** Zespół może śledzić postęp, pracować indywidualnie na gałęziach (ang. branches) i łączyć zmiany z główną wersją.
- **Rozproszony system kontroli wersji (DVCS):** W systemie rozproszonym nie ma scentralizowanego przechowywania plików. Istnieje wiele kopii zapasowych tego samego projektu.

In case of fire

1. git commit

2. git push

3. leave building

Git - najważniejsze komendy

- **git init:** Tworzy repozytorium Git w lokalnym katalogu.
- **git clone <adres-zdalnego-repozytorium>:** Kopiuje całe repozytorium z serwera zdalnego do lokalnego katalogu. Można również użyć tego polecenia do kopiowania lokalnych repozytoriów.
- **git add <plik.txt>:** Dodaje pojedynczy plik lub wiele plików i folderów do obszaru stagowania (ang. staging area).
- **git commit –m "Wiadomość"**: Tworzy migawkę zmian i zapisuje ją w repozytorium. Wiadomość służy do opisania wprowadzonych zmian.

Git - najważniejsze komendy

- **git config**: Używane do ustawiania konfiguracji użytkownika, takich jak adres e-mail, nazwa użytkownika i format pliku.
- **git status**: Pokazuje listę zmienionych plików lub plików, które nie zostały jeszcze dodane do obszaru stagowania i zatwierdzone.
- **git push <nazwa-zdalnego> <nazwa-gałęzi>**: Wysyła lokalne zatwierdzenia do zdalnej gałęzi repozytorium.

Git - najważniejsze komendy

- **git checkout -b <nazwa-gałęzi>**: Tworzy nową gałąź i przełącza się na nią.
- **git remote –v**: Wyświetla wszystkie zdalne repozytoria.
- **git remote add <nazwa-zdalnego> <adres-hosta-lub-URL-zdalnego>**: Dodaje serwer zdalny do lokalnego repozytorium.
- **git branch –d <nazwa-gałęzi>**: Usuwa gałąź.

Git - najważniejsze komendy

- **git pull**: Łączy zatwierdzenia z zdalnego repozytorium z lokalnym katalogiem.
- **git merge <nazwa-gałęzi>**: Po rozwiązaniu konfliktów scalania, polecenie łączy wybraną gałąź z bieżącą.
- **git log**: Pokazuje szczegółową listę zatwierdzeń dla bieżącej gałęzi.

git fork - kopia repozytorium

- Forkowanie repozytorium pozwala na swobodne eksperymentowanie z zmianami bez wpływu na oryginalne repozytorium.
- **Tworzenie własnej kopii**: Fork repozytorium tworzy osobistą kopię projektu na Twoim koncie GitHub (lub innym serwisie hostującym Git). Możesz robić zmiany w tej kopii bez wpływu na oryginalne repozytorium.

Git w praktyce

- Każdy po zajęciach wrzuca co zrobił na swój github
- Materiały będą na zajecia2025

Wirtualne Środowisko Python

- Wirtualne środowisko to izolowana przestrzeń dla projektu Python, umożliwiająca zarządzanie zależnościami bez wpływu na globalne ustawienia.
- **Instalacja virtualenv:**
  - Użyj **pip install virtualenv** w terminalu do instalacji pakietu.
- **Tworzenie środowiska:**
  - Wykonaj **virtualenv nazwa_srodowiska**, gdzie **nazwa_srodowiska** to wybrana nazwa folderu dla środowiska.
- **Aktywacja:**
  - Windows: **.\nazwa_srodowiska\Scripts\activate**
  - Unix (Linux, macOS): **source nazwa_srodowiska/bin/activate**
  - Po aktywacji, wszystkie polecenia **python** i **pip** odnoszą się do środowiska wirtualnego.
- **Dezaktywacja:**
  - Wykonaj **deactivate** aby powrócić do globalnego środowiska.