

WPROWADZENIE DO GIT

Krzysztof Morcinek i Tomasz Rusek
devWarsztaty
19 maja 2018

inspiracja - github.com/SkillsTemple/git-devWarsztaty

CZYM JEST GIT



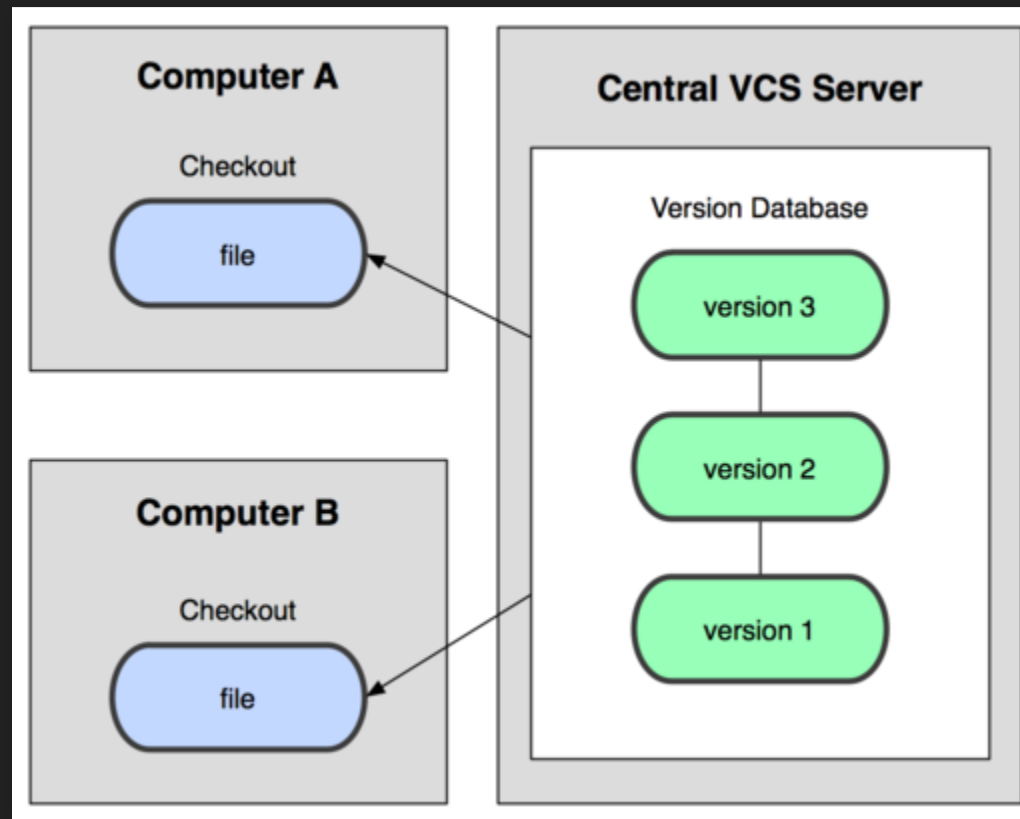
<https://xkcd.com/1597/>

GIT TO W ZASADZIE BAZA DANYCH

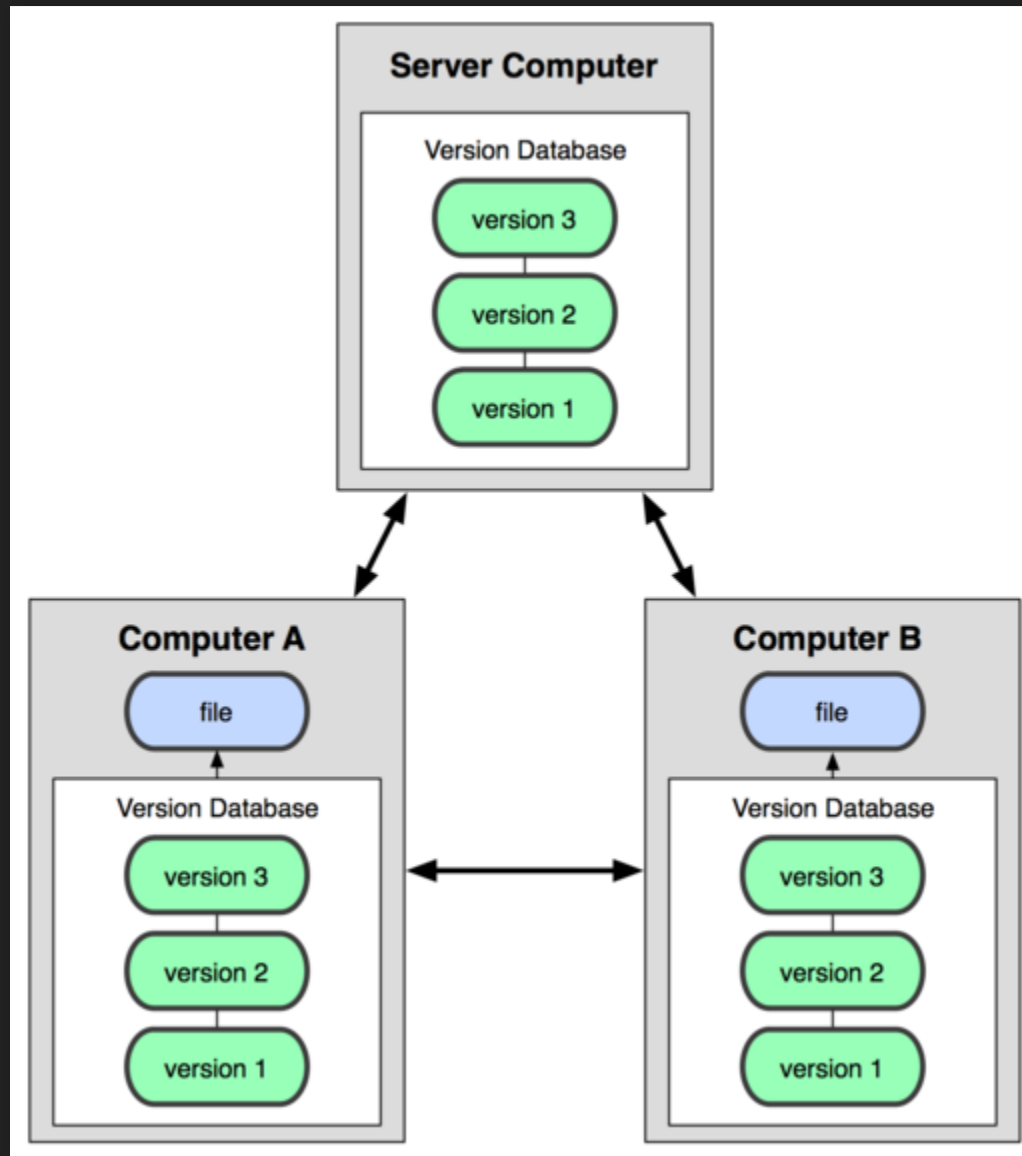
Git to VCS na który można patrzeć, jak na rozproszoną bazę danych, opartą na streamie snapshotów plików



Git **nie** jest scentralizowanym systemem kontroli wersji



Git jest zdecentralizowanym systemem kontroli wersji



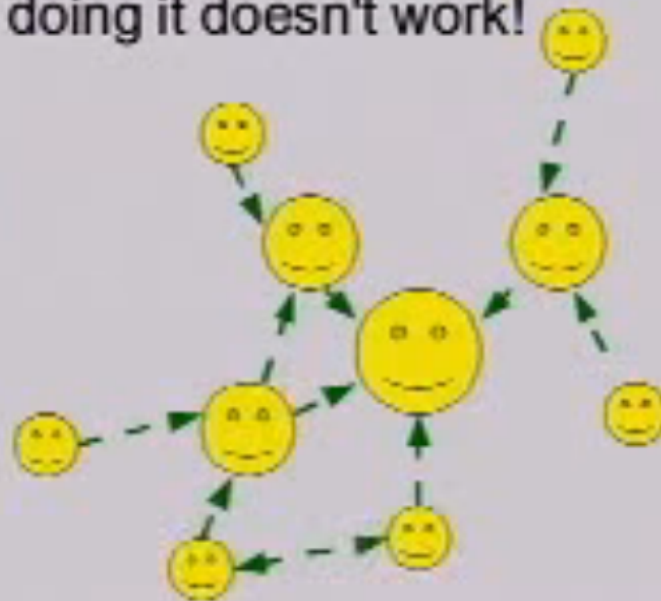
Git jest zdecentralizowanym systemem kontroli wersji

Distribution

(Look, ma, really cheesy graphics)

It's not just a good idea.

Not doing it doesn't work!



DLACZEGO GIT JEST GIT

- jest szybki
- jest rozproszony
- ułatwia rozwiązywanie konfliktów
- wspiera nieliniowy development (branche)
- działa offline
- pozwala na pracę nad jakością commitów
- bardzo łatwy do użytku domowego
- jest bezpieczny

JEST BEZPIECZNY - JEST WIARYGODNY

Suma kontrolna każdego commita opiera się na

- Zawartości i nazwach wszystkich plików
- ID parent commit(ów)
- Wiadomości (opisie) commita
- Autorze i/lub commiterze

Git się zorientuje przy nieprawidłowości w danych
(np. po uszkodzeniu dysku albo próbie sabotażu)

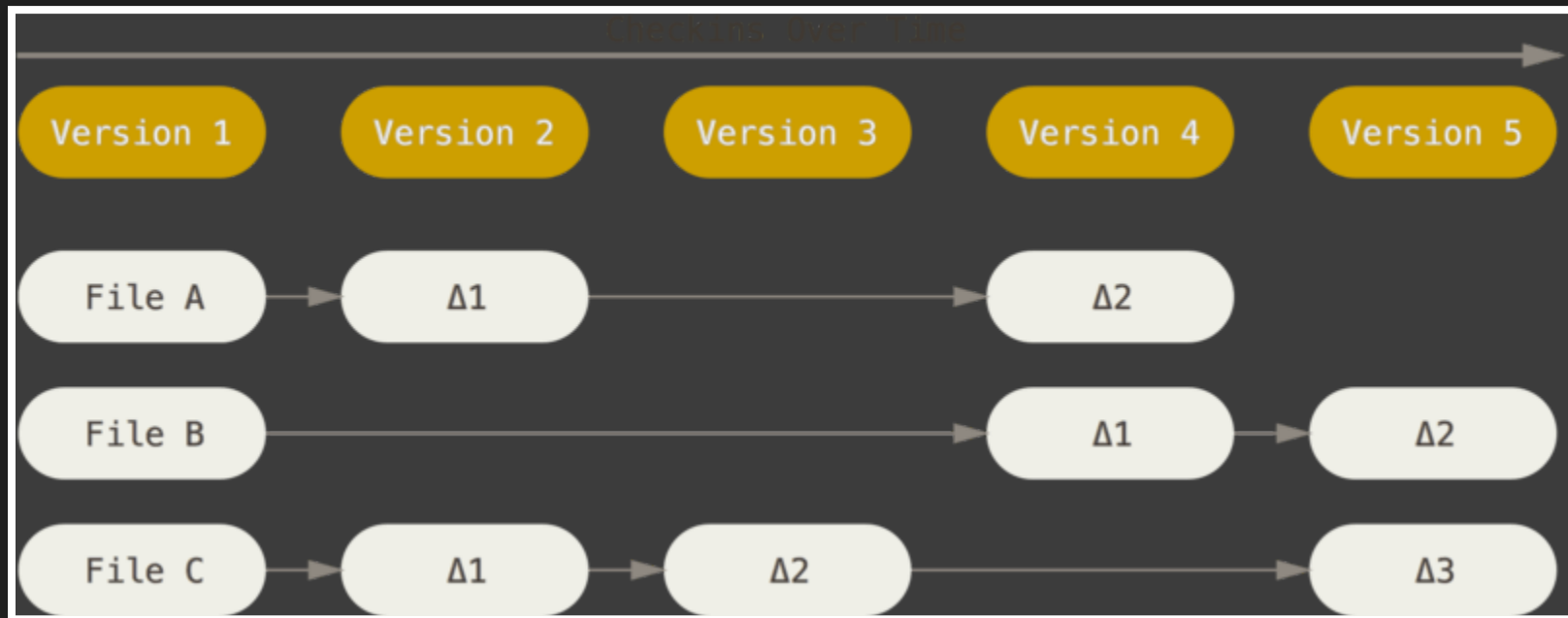
Analogicznie wiarygodne są przelewy w bitcoin

JAK DZIAŁA GIT

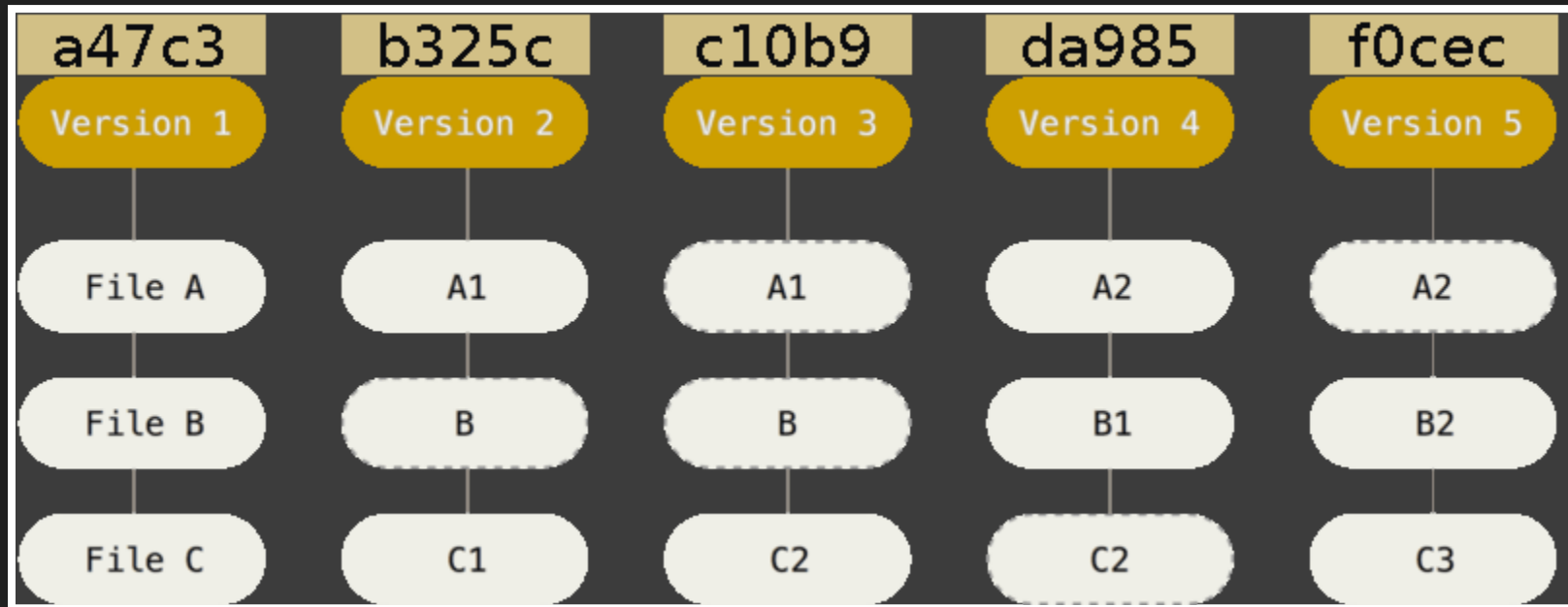
Git w podstawowym scenariuszu jedynie dodaje kolejne snapshoty plików



W svn commit to diff zmienionych plików (różnica)



W git commit to snapshot **wszystkich** plików



reprezentowany za pomocą hasha SHA-1

Dzięki temu, można wskazać dowolny commit i podejrzeć cały stan systemu z ówczas, a nie tylko jakie diffy ten commit ze sobą niósł.



Wiedząc jaka jest wersja na produkcji, zawsze można wrócić do niej w kodzie i zrobić drobną poprawkę bez wprowadzania na produkcję kolejnych commitów.

W PRZECIWIENSTWIE DO KRYPTOWALUT...

w swojej 'instacji bazy' (na swoim komputerze) możesz zmieniać istniejące snapshoty (commity)

ALE!

tylko pod warunkiem, że jeszcze ich nie opublikowałeś
innym bazom

GIT NIE ŚLEDZI PLIKÓW

W przeciwieństwie do niektórych VCS, git nie śledzi plików, wyłącznie ich zawartość

```
on branch gh-pages  
Your branch is ahead of 'origin/gh-pages' by 1 commit.  
(use "git push" to publish your local commits)  
Changes to be committed:  
(use "git reset HEAD <file>..." to unstage)
```

```
renamed:    index.html -> main.html
```

GIT wyświetlając informację o rename pliku, tak naprawdę jedynie domyśla się, co człowiek zrobił - dla niego jedynie część danych się przeniosła, doszedł nowy plik, a jeden istniejący zniknął

HEAD

HEAD to referencja do obecnie zcheckoutowanego commita, zazwyczaj będącego ostatnim commitem w branchu

- HEAD - ostatni commit (*pierwszy od końca*) - np. *ed8ab42[...]*
- HEAD~1 - przedostatni commit - np. *daa6b92[...]*
- HEAD^ - przedostatni commit - np. *a6bc32b[...]*
- HEAD~3 - czwarty od końca commit - np. *7cb16f7[...]*
- HEAD^^^ - czwarty od końca commit - np. *343ebb3[...]*

Pytania?

WORKING DIRECTORY, STAGING AREA, COMMIT

Twoje pliki - absolutnie każdy plik który utworzysz/edytujesz/usuniesz, zawsze jest w którymś z 3 "obszarów" gita

- obszarze roboczym (working directory)
- 'indeksie' (staged files)
- repozytorium (commicie dodanym do historii)

OBSZAR ROBOCZY

Obszar roboczy (working directory) to obszar w którym jest praca którą wykonałeś, ale o której jeszcze nie powiedziałeś nic gitowi

- nowe pliki których wcześniej nie commitowałeś
- zmiany w plikach które wcześniej commitowałeś
- zmiana nazwy pliku lub jego usunięcie

Git nie wie nic o Twojej pracy w working directory, uważaj żeby jej nie stracić dopóki mu o niej nie powiesz

Zmiany w working directory odnoszą się do staging area, a jeśli danego pliku nie ma w staging area, wówczas bezpośrednio do obecnej wersji w lokalnym repozytorium

INDEX (STAGED FILES)

Staging area to obszar w którym przygotowujesz sobie które zmiany zostaną zacomitowane, czyli dodane do Twojego lokalnego repozytorium

Po zastagowaniu zmiany, czyli przygotowaniu jej do commita, dalej można plik edytować, wówczas zmiany w working directory pokazywane są względem zastagowanej wersji

Co więcej, od razu można wybrać żeby zastagować tylko część zmian

Dla przykładu na dole pliku zaczęliśmy dopisywać nowy kod, a w środku pliku zobaczyliśmy literówkę w tekście. Wówczas można od razu nieprzerywając pracy na dole pliku, zastagować jedynie poprawienie literówki i dodać commita "Fix typo in module A".

COMMIT

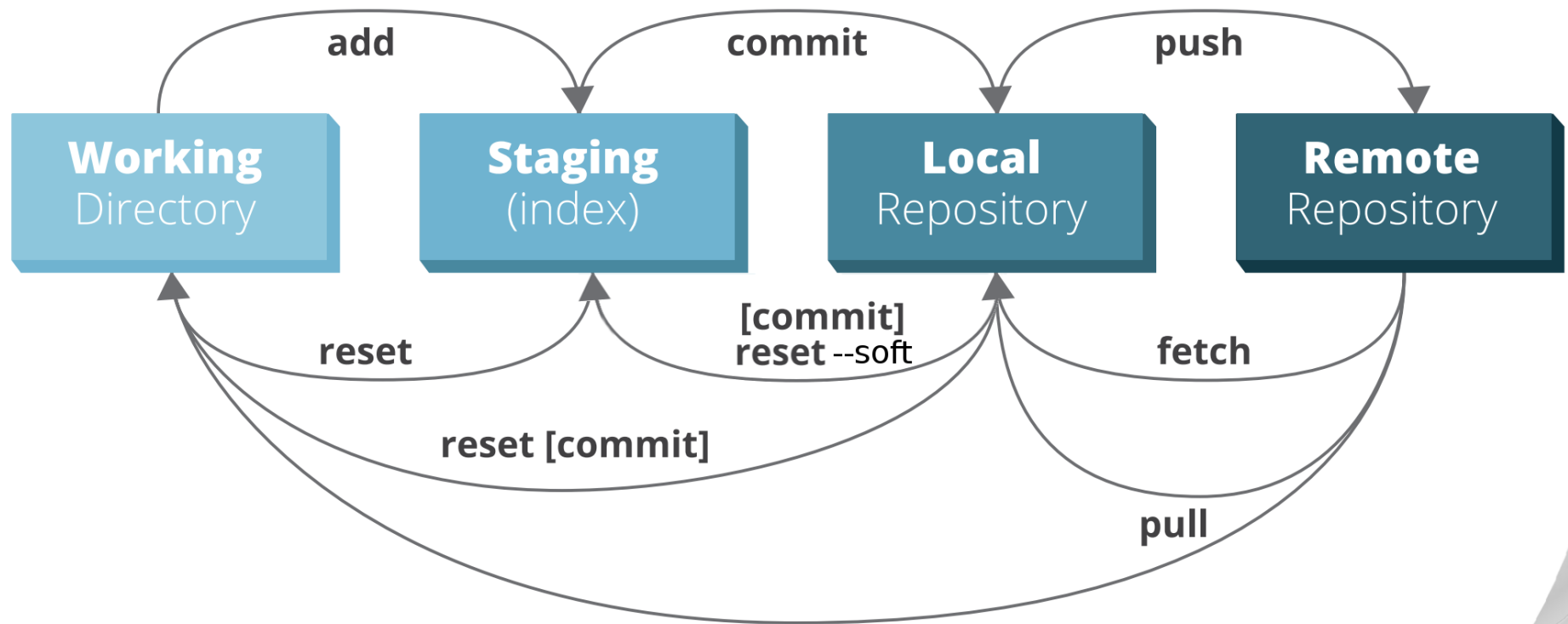
Snapshot wszystkich plików w projekcie, posiadający identyfikator SHA-1

```
commit 7cb1df774081fc1b9d0c97c262cbefc202d79ffa
Author: Tometchy <tometchy@gmail.com>
Date:   Wed May 16 20:36:23 2018 +0200

    Add info that git doesn't track files

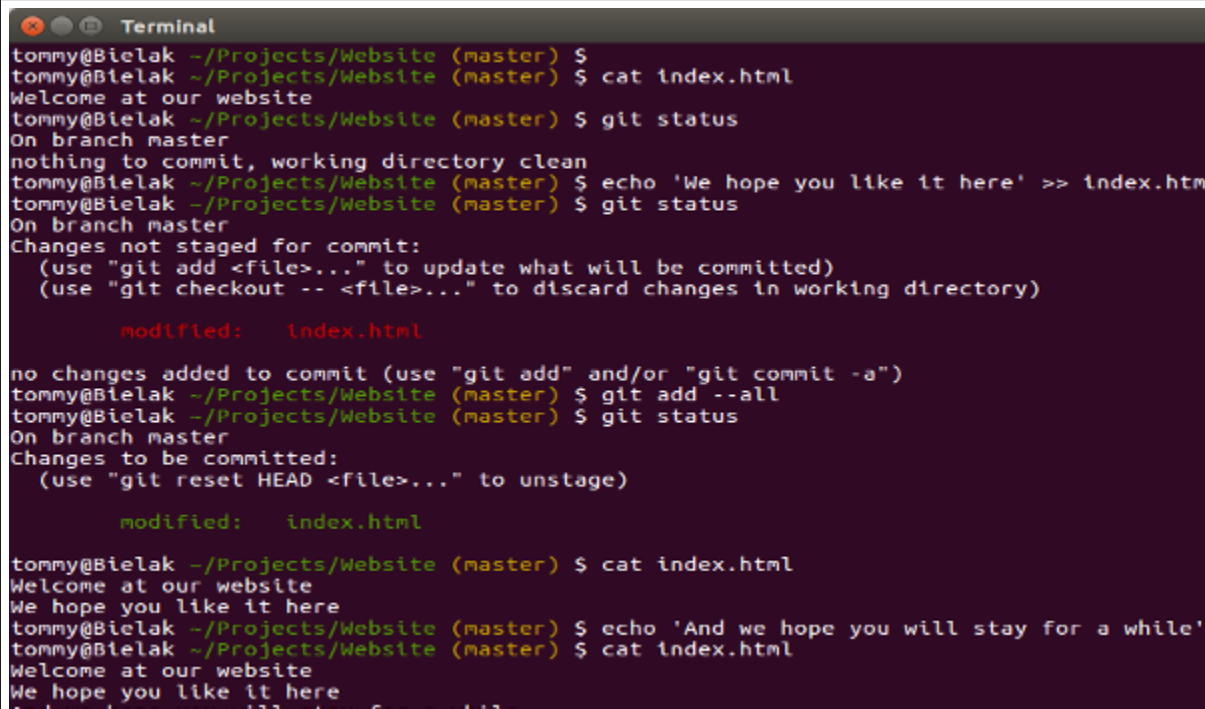
slides/drafts/git-doesnt-track.xcf | Bin 0 -> 109979 bytes
slides/img/git-doesnt-track.png   | Bin 0 -> 32647 bytes
slides/index.html                 |    5 +++++
3 files changed, 5 insertions(+)
```

FLOW



PRZYKŁAD STAGOWANIA (GIT ADD)

git-work-example.png

A terminal window titled "Terminal" showing a series of commands and their outputs. The user is in a directory ~/Projects/Website on the master branch. They first create an index.html file with the text "Welcome at our website". Then they run 'git status', which shows the directory is clean. Next, they append a new line to index.html: "We hope you like it here". Running 'git status' again shows that index.html is modified but not staged. The user then runs 'git add --all', which stages the change. A second 'git status' shows the change is staged for commit. Finally, they run 'cat index.html' again, showing the full content of the file.

```
tommy@Bielak ~/Projects/Website (master) $  
tommy@Bielak ~/Projects/Website (master) $ cat index.html  
Welcome at our website  
tommy@Bielak ~/Projects/Website (master) $ git status  
On branch master  
nothing to commit, working directory clean  
tommy@Bielak ~/Projects/Website (master) $ echo 'We hope you like it here' >> index.htm  
tommy@Bielak ~/Projects/Website (master) $ git status  
On branch master  
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:   index.html  
  
no changes added to commit (use "git add" and/or "git commit -a")  
tommy@Bielak ~/Projects/Website (master) $ git add --all  
tommy@Bielak ~/Projects/Website (master) $ git status  
On branch master  
Changes to be committed:  
  (use "git reset HEAD <file>..." to unstage)  
  
        modified:   index.html  
  
tommy@Bielak ~/Projects/Website (master) $ cat index.html  
Welcome at our website  
We hope you like it here  
tommy@Bielak ~/Projects/Website (master) $ echo 'And we hope you will stay for a while'  
tommy@Bielak ~/Projects/Website (master) $ cat index.html  
Welcome at our website  
We hope you like it here  
And we hope you will stay for a while
```

Pytania?

PRZYKŁADOWE FLOW

- Demo pokazywanie statusu w SourceTree

ĆWICZENIE

1. Wejdź do nowo utworzonego katalogu
2. Utwórz repozytorium - `git init`
3. Stwórz plik `index.txt` z zawartością "Hello World"
4. Dodaj plik do repozytorium (`git add`, `git commit`)
5. Zmień treść pliku na "Witaj świecie"
6. Sprawdź status repozytorium (`git status`)
7. Dodaj plik do repozytorium (`git add`, `git commit`)
8. Dodaj do pliku kolejną linię "jest git"
9. ...

CD ĆWICZENIE

- ...
- Stwórz nowy plik cars.json i wpisz "toyota"
- Sprawdź status repozytorium (git status)
- Dodaj plik cars.json do repozytorium (git add *ścieżka*, git commit)
- Sprawdź status repozytorium (git status)
- "Skomituj" plik index.txt
- Usuń plik **cars.json**
- Sprawdź status repozytorium (git status)
- "Skomituj" usunięcie pliku

PRZEGLĄDANIE HISTORII

- git log
- gitk

GIT LOG FOR EXAMPLE WITH GRAPH

```
Terminal
tommy@Bielak ~/Desktop/temp/material-components-android (master) $ git log --graph --decorate
* commit 768659d5f861efe278c96947dfa29023895a6970 (HEAD -> master)
  Author: Tometchy <tometchy@gmail.com>
  Date:   Fri May 18 22:03:32 2018 +0200

    Update readme file

*   commit 2478632c421929ef1bf96ca1ad8b48e3211d2277
  |\
  | Merge: 7fa43fc 8261465
  | Author: Tometchy <tometchy@gmail.com>
  | Date:   Fri May 18 22:02:19 2018 +0200
  |
  | Merge branch 'travisci'
  |
*   commit 8261465a10a0289f4b1e717038760c1a36662091 (origin/travisci, travisci)
  |\
  | Author: Cameron Ketcham <cketcham@google.com>
  | Date:   Thu May 17 17:08:42 2018 -0400
  |
tommy@Bielak ~/Desktop/temp/material-components-android (master) $
```

GIT LOG FOR EXAMPLE IN ONE LINE

```
Terminal
| * 53ba33d Fix android-P repo
| * d8b646b Remove flaky test
tommy@Bielak ~/Desktop/temp/material-components-android (master) $ git log -5 --oneline
392aeca Update readme file
2478632 Merge branch 'travisci'
7fa43fc Add null check before validating attribute set to avoid NPE.
1cad400 Roll forward box background fix.
eb89b76 Prevent user from setting start/end compound drawable on Chips. They should be set via app:chipIcon
and app:closeIcon instead.
tommy@Bielak ~/Desktop/temp/material-components-android (master) $ git log -5 --oneline --graph
* 392aeca Update readme file
* 2478632 Merge branch 'travisci'
| \
| * 8261465 Add lynx back
| * 53ba33d Fix android-P repo
tommy@Bielak ~/Desktop/temp/material-components-android (master) $ git log -5 --oneline --graph --decorate
* 392aeca (HEAD -> master) Update readme file
* 2478632 Merge branch 'travisci'
| \
| * 8261465 (origin/travisci, travisci) Add lynx back
| * 53ba33d Fix android-P repo
| * d8b646b Remove flaky test
tommy@Bielak ~/Desktop/temp/material-components-android (master) $
```

GIT LOG FOR EXAMPLE SHOWING STATS AND PATCH

```
Terminal
tommy@Bielak ~/Desktop/temp/material-components-android (master) $ git log -1 --stat --patch
commit 392aecab5aff1467bc329af9b60d02e7e6a94915
Author: Tometchy <tometchy@gmail.com>
Date:   Fri May 18 22:03:32 2018 +0200

    Update readme file
---
 README.md | 2 ++
 1 file changed, 2 insertions(+)

diff --git a/README.md b/README.md
index b2e8da6..3de54ea 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,5 @@
+# Brand new section
+Brand new description
+[[Build Status]](https://img.shields.io/travis/material-components/material-components-android/master.svg)(https://travis-ci.org/material-components/material-components-android/)
+[[Chat]](https://img.shields.io/discord/259087343246508035.svg)(https://discord.gg/material-components)

tommy@Bielak ~/Desktop/temp/material-components-android (master) $
```

GITK

The screenshot shows the GitK application window titled "material-components-android: All files - gitk". The window has a menu bar with "File", "Edit", "View", and "Help".

The main area is divided into three panes:

- Left Pane (Commit History):** A vertical list of commits. The current commit is highlighted in green and labeled "master". The commit message is "Update readme file". Below it, a merge branch "travisci" is shown. The commit is linked to "remotes/origin/travisci" and "travisci". The commit message is "Add lynx back". Other commits in the history include "Fix android-P repo", "Remove flaky test", "Modify travis build to cache android sdk", "Remove flaky test", "Remove flaky test", "Only run one set of tests", "Remove tests", and "Add findbugs dep for tests".
- Right Pane (Commit Log):** A table showing the commit log for the selected commit. It includes the commit hash, the committer's name and email, and the commit date and time.
- Bottom Pane (Commit Details):** A section for the selected commit, showing the SHA1 ID, the commit message, and the commit details.

The SHA1 ID is displayed as "d8b646bb3842161879bbcf7a0f9231580dba4b9e". The commit message is "Remove flaky test".

Below the commit details, there is a "Find" section with a search bar and a "Search" button. The search bar contains the text "commit containing:". The "Search" button is labeled "Search".

At the bottom, there is a "Diff" section with tabs for "Diff", "Old version", and "New version". The "Diff" tab is selected. The diff shows the changes in the file "tests/javatests/com/google/android/material/textfield/TextInputLayoutTest.java". The diff includes the commit message "Remove flaky test".

Commit Hash	Committer	Date
d8b646bb3842161879bbcf7a0f9231580dba4b9e	Tometchy <tometchy@gmail.com>	2018-05-18 22:03:32
...
...	Tometchy <tometchy@gmail.com>	2018-05-18 22:02:19
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 23:08:42
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 22:50:12
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 22:45:16
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 22:40:47
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 22:28:04
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 22:13:38
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 21:50:27
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 19:36:30
...	Cameron Ketcham <cketcham@google.com>	2018-05-17 17:52:56

SHA1 ID: d8b646bb3842161879bbcf7a0f9231580dba4b9e

Find: commit containing: [Search]

Diff: Remove flaky test

Comments: tests/javatests/com/google/android/material/textfield/TextInputLayoutTest.java

SZYBKIE ĆWICZENIE

Przejrzyj historię commitów za pomocą git log i gitk

- różne wariacje git log

- -5
- --graph
- --oneline
- --decorate
- --pretty-print

- gitk

- gitk
- gitk --all
- gitk master

EDYTOWANIE ISTNIEJĄCYCH ZMIAN

- Przed opublikowaniem innym
- Po opublikowaniu innym

EDYCJA 'OSTATNICH' COMMITÓW PRZED OPUBLIKOWANIEM

- `git reset`
 - `git reset --mixed`
 - `git reset --soft`
 - `git reset --hard`
- `commit --amend`
- `rebase --interactive`

EDYCJA 'OSTATNICH' COMMITÓW PO OPUBLIKOWANIU

`git revert`

np:

- `git revert HEAD`
- `git revert HEAD~4`
- `git revert 6e5av3a`

DEMO REBASE --INTERACTIVE

- `git rebase --interactive HEAD~4`
- `git rebase -i develop`

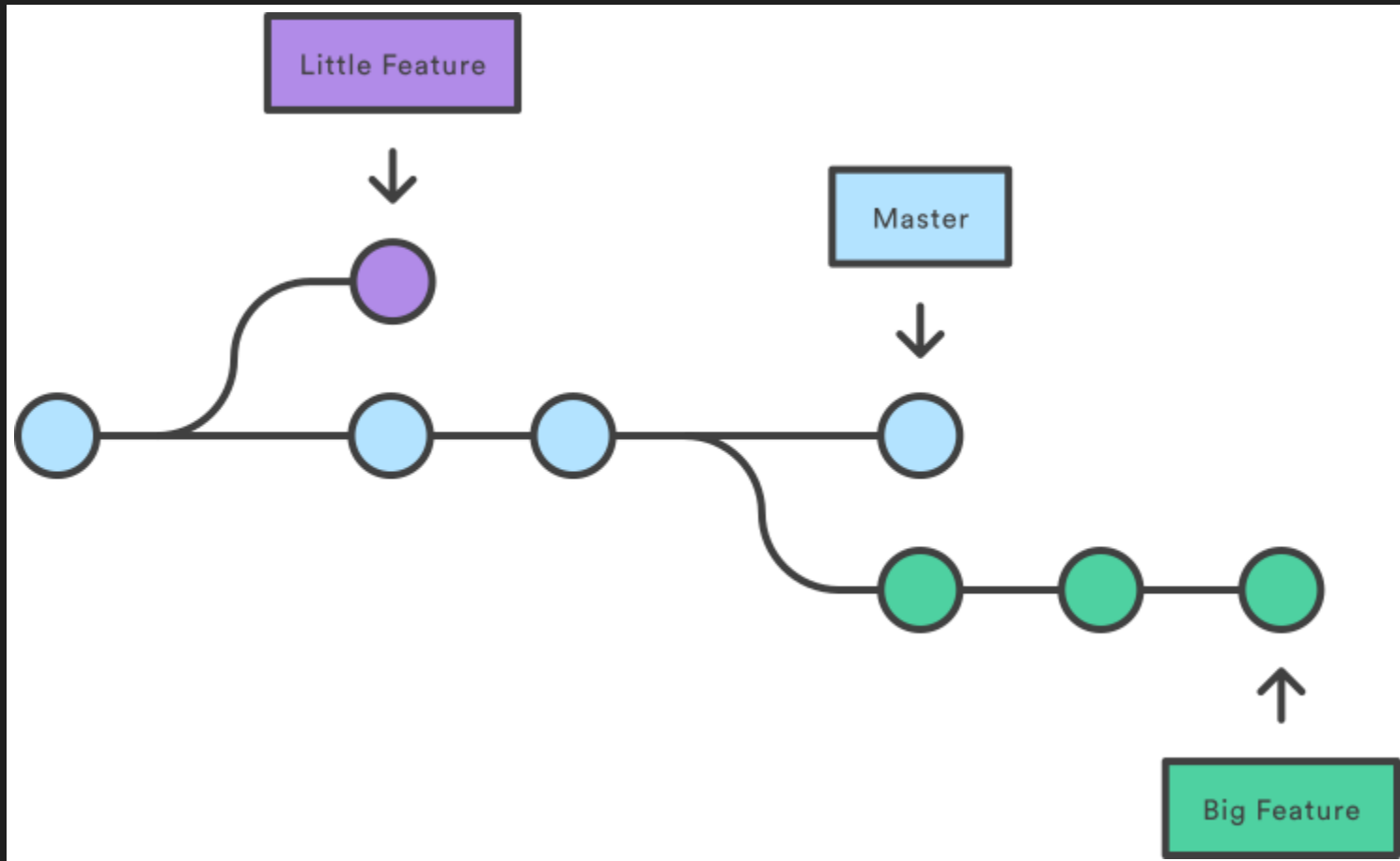
ĆWICZENIE 1 (OPIS W REPO)

`git rebase --interactive`

ĆWICZENIE 2 - 'ZABAWA' Z RESETOWANIEM COMMITÓW I OBSERWACJA 'CO SIĘ DZIEJE'

```
git reset --soft/mixed/hard HEAD~LICZBA_COMMITÓW
```

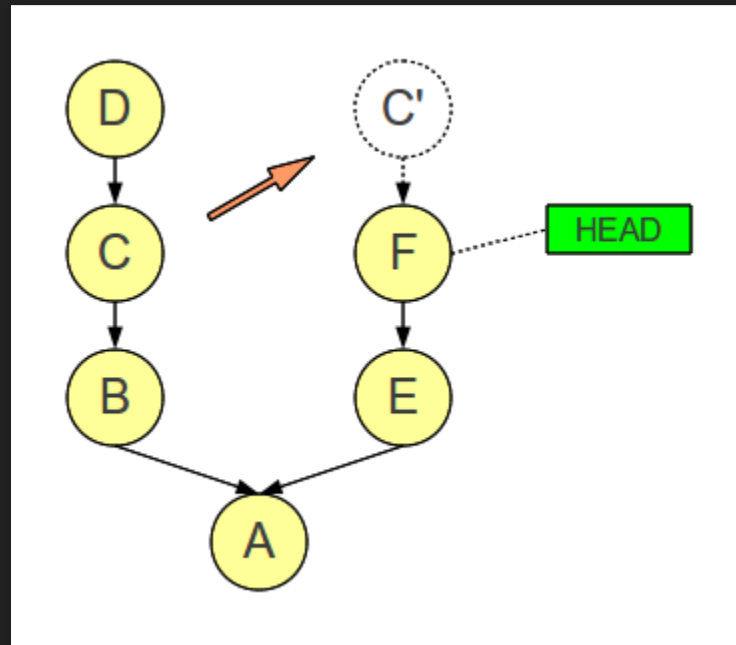
BRANCHE



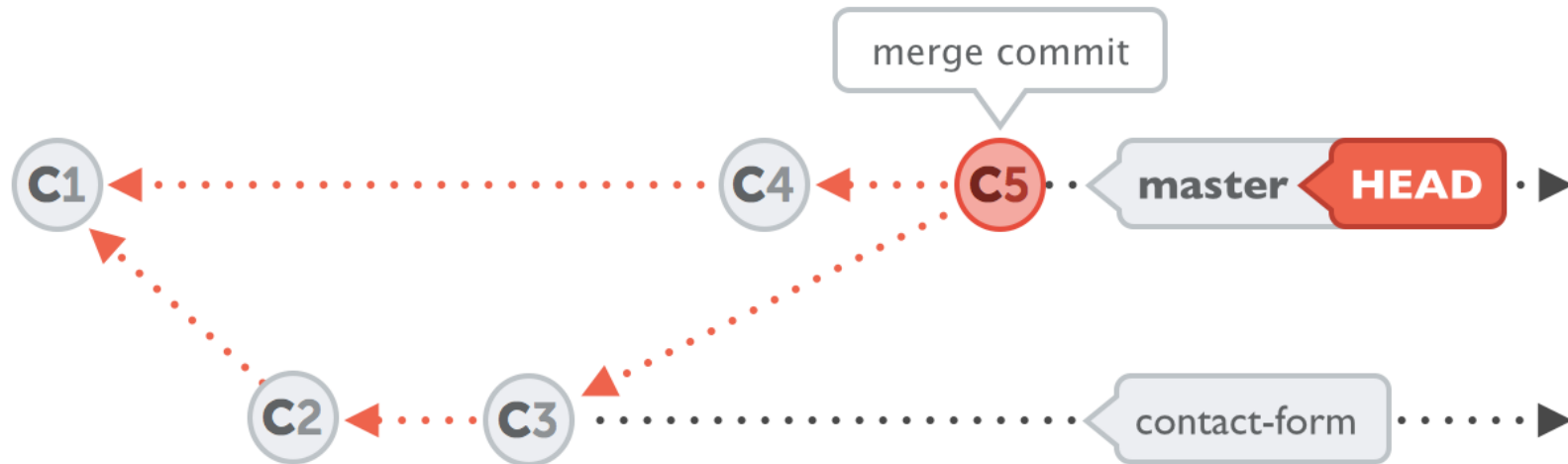
Learning branching

CHERRY-PICK

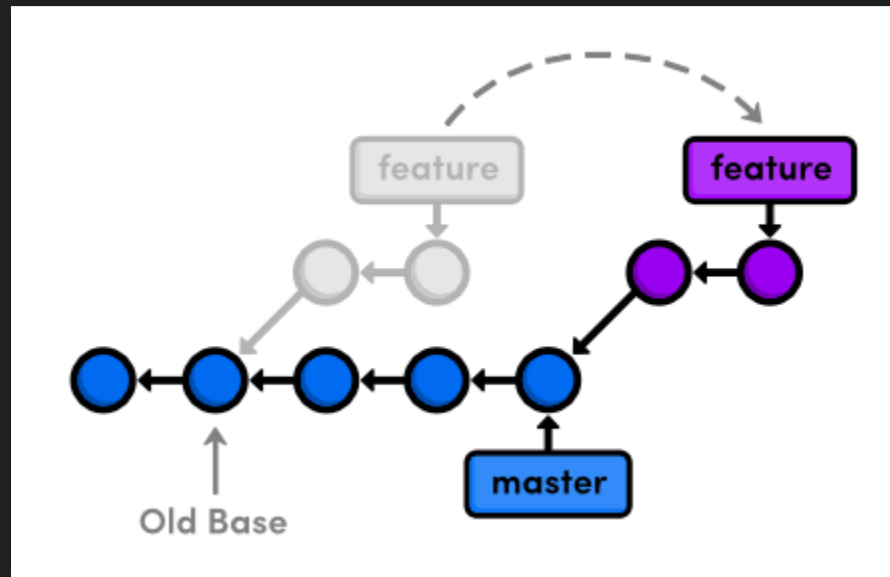
Commity między branchami można przekładać



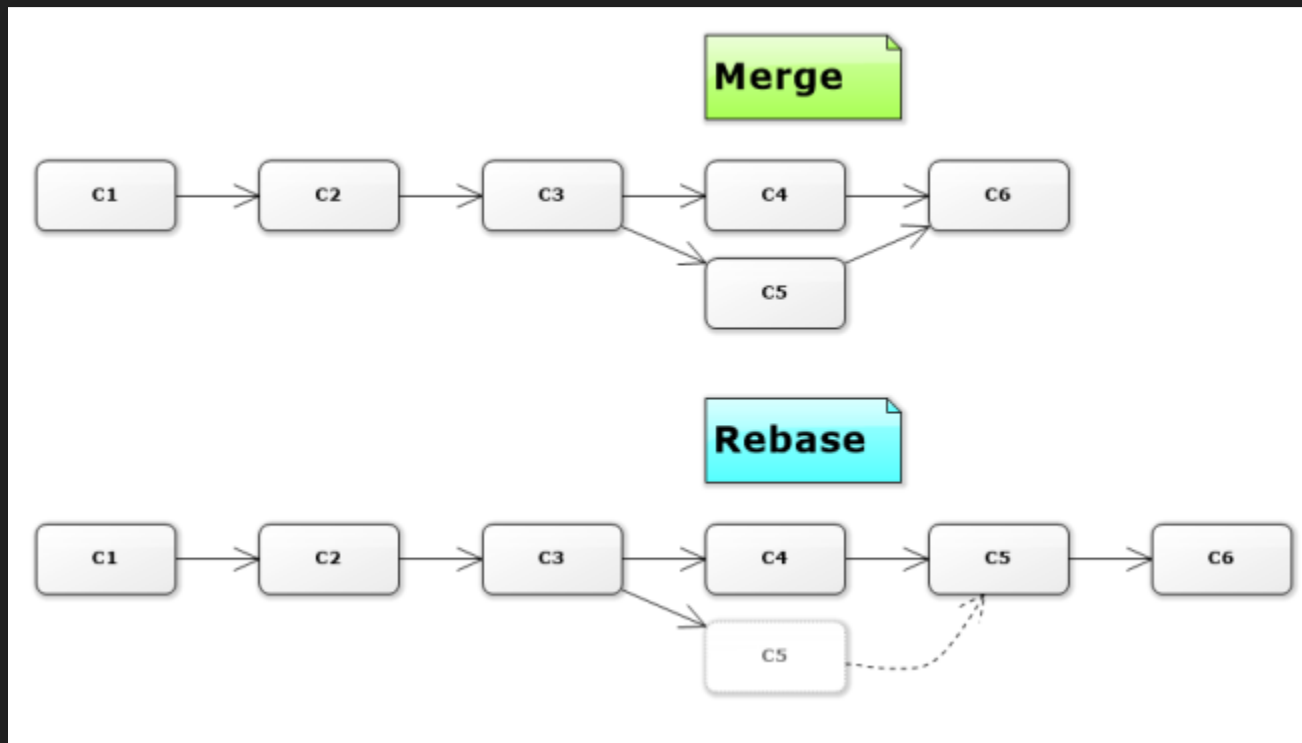
MERGE



REBASE



MERGE VS REBASE



1. Ćwiczenie pierwsze - zrób merge lub rebase
2. Ćwiczenie drugie - rozwiąż konflikty

REBASE TO ŚWIETNE NARZĘDZIE...

Rebase to świetne narzędzie, poprawianie czytelności historii bardzo ułatwia przeglądanie repozytorium.

Ale trzeba być ostrożnym. Czasami warto przerwać rebasowanie i jednak zrobić zwykłego merge.

[link do fajnego artykułu tłumaczącego trudności które można napotkać](#)

- rebase zakłamuje historię (z dobrą intencją, ale jednak)
- rebase może wywołać konflikty których by nie było przy mergu
 - a w związku z tym, może wprowadzić błąd na produkcję
 - pomijając, że to dokłada dodatkowej roboty

PUSHOWANIE & PULLOWANIE ZMIAN

Synchronizacja repozytoriów (np. lokalnego z wersją na GitHubie) odbywa się za pomocą 'wypychania' i 'ciągnięcia' commitów

- git push
- git pull
 - git fetch
 - git merge

PULL Z REBASE

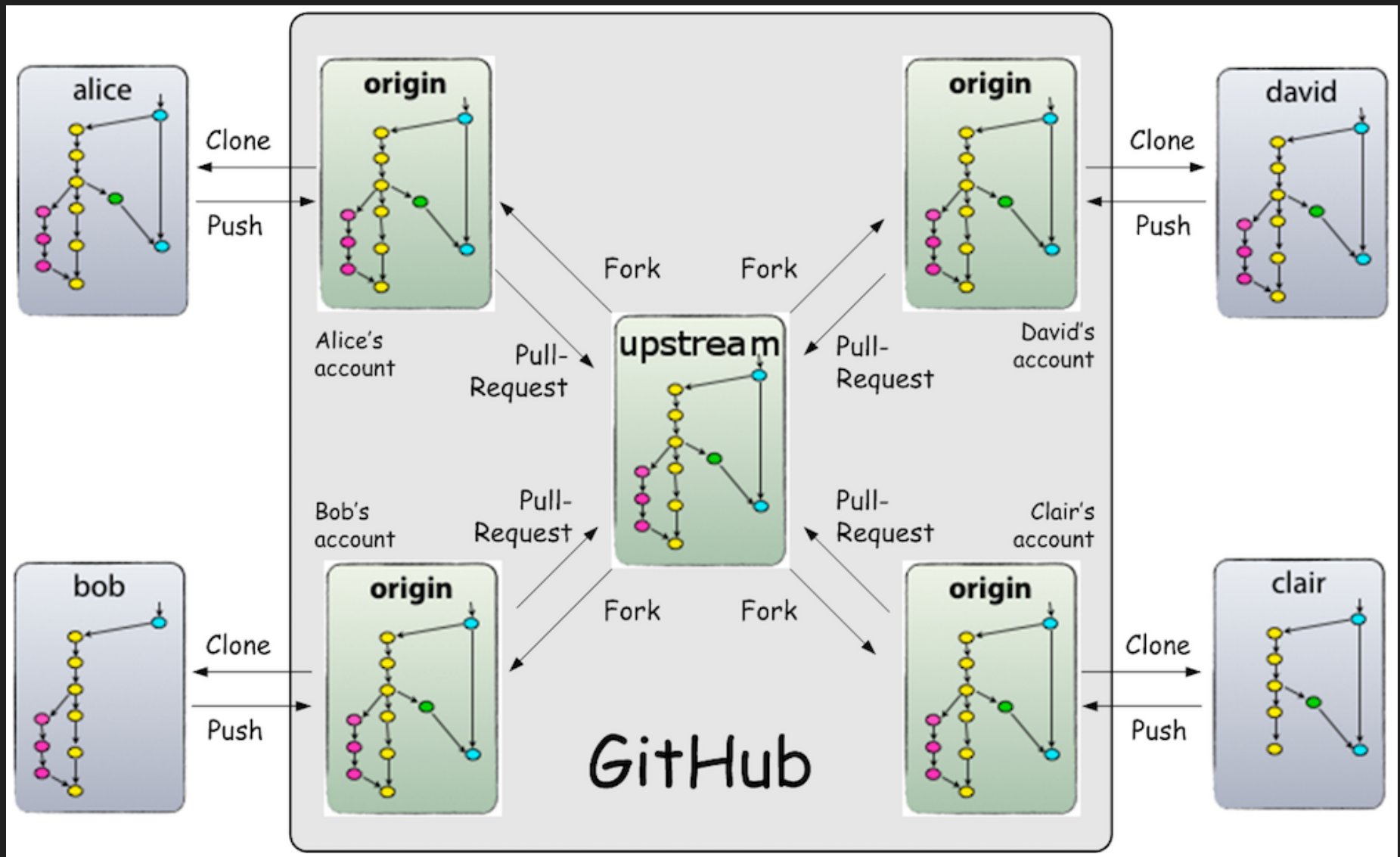
Często dobrą opcją jest zastosowanie komendy która od razu za nas zrebasuje commity

```
git pull --rebase
```

GIT REFLOG

```
jekyll master $ git reflog
6703083... HEAD@{0}: pull origin master: Fast forward
fe71d2b... HEAD@{1}: commit: Updating README with added
eac6b03... HEAD@{2}: commit: Making sure that posts fla
f682c8f... HEAD@{3}: commit: Added publish flag to post
9478f1b... HEAD@{4}: rebase: Modifying the README a bit
7e178ff... HEAD@{5}: checkout: moving from master to 7e
cda3b9e... HEAD@{6}: HEAD~1: updating HEAD
3b75036... HEAD@{7}: merge newfeature: Merge made by re
cda3b9e... HEAD@{8}: commit: Modifying the README a bit
6981921... HEAD@{9}: checkout: moving from newfeature t
7e178ff... HEAD@{10}: commit: Rewriting history is fun
4a97573... HEAD@{11}: commit: all done with TODOs
eb60603... HEAD@{12}: commit: README
```


FORKI & PULL REQUEST



PYTANIA? - PULL REQUESTY!

PRZYDATNE NARZĘDZIA GITHUBA ZWIAŹANE Z GITEM

- gist
- GitHub Pages

LINKI

- [Pro Git book \(Scott Chacon, Ben Straub\)](#)
- [Atlassian Tutorials](#)
- [Learn git branching](#)
- [Visualise git with D3](#)
- [Successful git branching model - Git flow](#)
- [Oh shit, git!](#)
- [Git cheatsheet](#)
- [How to undo \(almost\) anything with git](#)
- [How to Write a Git Commit Message](#)
- [GIT Illustrated Cheatsheet](#)
- [Artykuł na co uważać przy rebase](#)

DZIĘKUJEMY!

```
Terminal
Thank you for your attention!

It was pleasure :)

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#   modified:   thank.you
#
~
~
~
~
~
1 line less; before #6  4 seconds ago          13,1          All
```