

# WPROWADZENIE DO GIT

Krzysztof Morcinek i Tomasz Skraskowski

Dev Warsztaty  
Katowice - 12 stycznia 2019

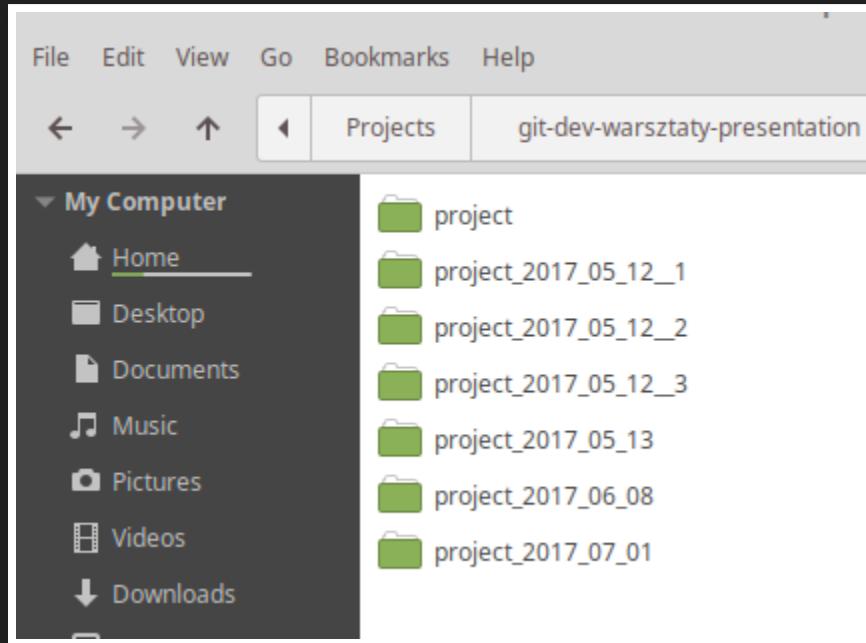
Poprzednie edycje:  
Wrocław - 19 maja 2018

*inspiracja - [github.com/SkillsTemple/git-devWarsztaty](https://github.com/SkillsTemple/git-devWarsztaty)*

# KWESTIE ORGANIZACYJNE

- Prośba o zalogowanie się na kanał slackowy
- Informacyjnie - na końcu odbędzie się losowanie nagrody od sponsora

# VCS TO DOBRA RZECZ



Tak to się kiedyś robiło ;)

Polecamy korzystanie z systemu kontroli wersji nie tylko do programowania

# KONSOLA & GITK & KDIFF3



Wielu ludzi upiera się, że zarządzania gitem z konsoli nie spróbuje, że pisanie zamiast klikania to przeżytek

Ostatnio przeczytane

Krzysztof Porębski 16:37  
powiem Ci, że zmiana na konsolce

w gitcie mocno na plus



mozesz agitowac

podajac mnie jako przyklad

bo sie bronilem rekami i nogami

16:37  
Cieszy mnie to 😊

Wpisz nową wiadomość

A, C, 🎉, 😊, GIF, 📸, ⌂, ...

▶

# KONFIGURACJA ŚRODOWISKA



# TRZY POZIOMY KONFIGURACJI

- **--system** (brany pod uwagę na końcu)  
Linux: /etc/gitconfig  
  
Windows: %ProgramFiles(x86)%\Git\etc\gitconfig  
  
Windows: %ProgramFiles%\Git\mingw64\etc\gitconfig
- **--global** (brany pod uwagę w drugiej kolejności)  
Linux: ~/.gitconfig  
  
Windows: %USERPROFILE%\.gitconfig
- **--local** (brany pod uwagę w pierwszej kolejności)  
.git/config

# MODYFIKOWANIE KONFIGURACJI

- **git config --POZIOM SEKCJA.NAZWA "TREŚĆ"**  
Np. `git config --global user.name "John Doe"`
- **git config --POZIOM --edit**  
Np. `git config --global -e`

Domyślny poziom przy zapisywaniu to local

# ODCZYTYWANIE KONFIGURACJI

- W edytorze tekstowym: `git config (--POZIOM) --edit`  
Np. `git config --global -e`
- Konkretny wpis: `git config (--POZIOM) SEKCJA.NAZWA`  
Np. `git config --get user.email`
- Listowanie wpisów: `git config (--POZIOM) --list`  
Np. `git config --list`

Bez podania poziomu, git przyjmuje obecnie brany pod uwagę poziom

# RÓŻNE EDYTORY TEKSTOWE

- vim

```
git config --global core.editor vim
```

- Linux Mint - xed

```
git config --global core.editor xed
```

- Ubuntu - gedit

```
git config --global core.editor gedit
```

- Windows - notepad

```
git config --global core.editor notepad
```

- Windows - notepad++

```
git config --global core.editor "'C:/Program  
Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"
```

- Windows - Visual Studio Code

```
git config --global core.editor "'C:\Program Files\Microsoft VS  
Code\code.exe' -n -w"
```

# PRZYKŁADOWA KONFIGURACJA



A screenshot of a terminal window titled "Terminal". The window displays a git configuration file with the following content:

```
[user]
email = tometchy@gmail.com
name = Tometchy
[push]
default = simple
[credential]
#helper = cache --timeout=36000
helper = /usr/share/doc/git/contrib/credential/libsecret/git-credential-libsecret
[core]
editor = vim
[merge]
tool = kdiff3
[diff]
guitool = kdiff3
~
~
"~/gitconfig" 14L, 276C          2,1-8      All
```

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 zaccommutowane, czyli doddane 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

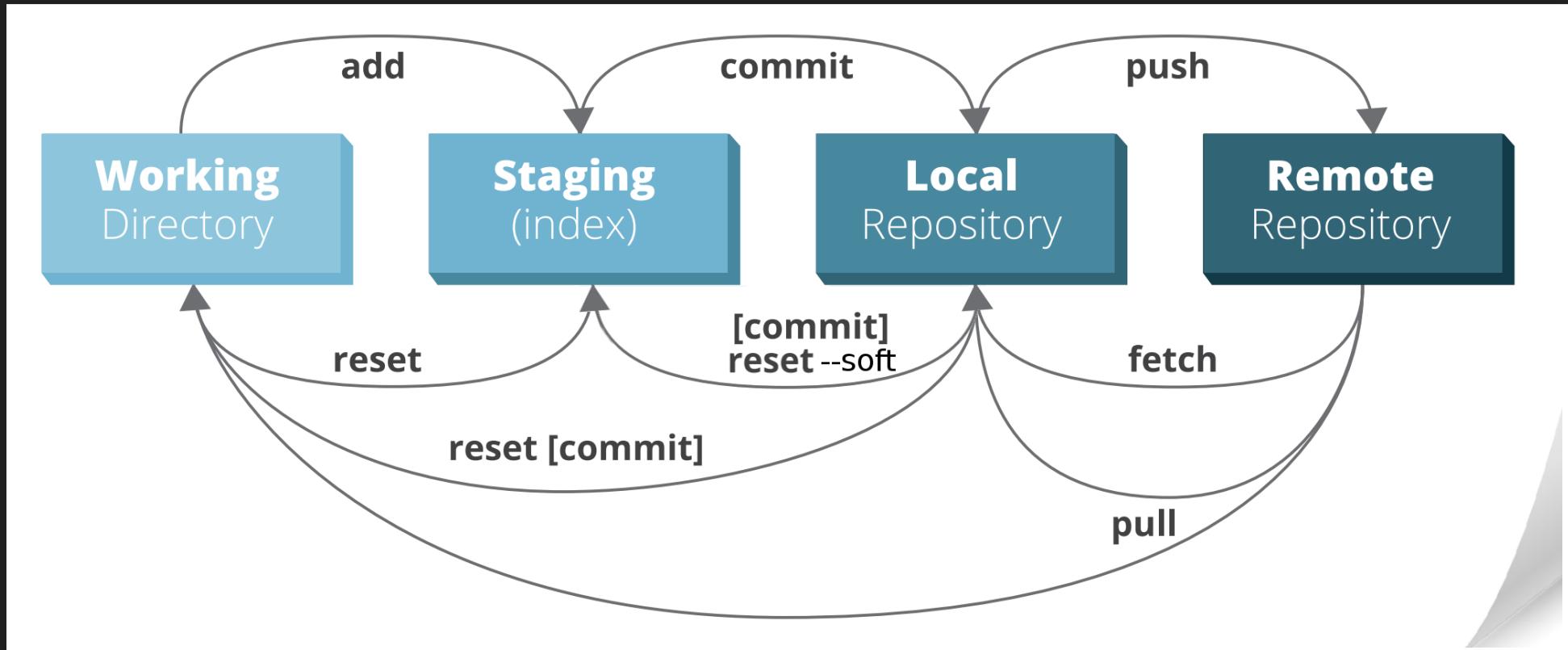
Ukazanie różnicy pomiędzy ostatnią wersją, a jednocześnie snapshot wszystkich plików w projekcie, posiadający identyfikator SHA-1.

```
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)



```
Terminal
tommy@Bielak ~/Projects/Website (master) $
tommy@Bielak ~/Projects/Website (master) $ cat index.html
Welcome at our website
```

# PRZYKŁAD STAGOWANIA (GIT ADD)



```
Terminal
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
```

# PRZYKŁAD STAGOWANIA (GIT ADD)



```
Terminal
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.html
```

# PRZYKŁAD STAGOWANIA (GIT ADD)



```
Terminal
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.html
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")
```

# PRZYKŁAD STAGOWANIA (GIT ADD)

```
Terminal
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.html
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
```



# PRZYKŁAD STAGOWANIA (GIT ADD)

```
Terminal
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.html
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
```



# PRZYKŁAD STAGOWANIA (GIT ADD)

```
Terminal
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.html
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
```



# PRZYKŁAD STAGOWANIA (GIT ADD)

```
Terminal
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' >> index.html
```

# PRZYKŁAD STAGOWANIA (GIT ADD)

```
Terminal
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' >> index.html
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
```



# PRZYKŁAD STAGOWANIA (GIT ADD)

```
Terminal
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' >> index.html
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
tommy@Bielak ~/Projects/Website (master) $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   index.html

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
```



# PRZYKŁAD STAGOWANIA (GIT ADD)

```
Terminal
We hope you like it here
And we hope you will stay for a while
tommy@Bielak ~/Projects/Website (master) $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   index.html

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

tommy@Bielak ~/Projects/Website (master) $ git diff
diff --git a/index.html b/index.html
index ded25b6..fb6438e 100644
--- a/index.html
+++ b/index.html
@@ -1,2 +1,3 @@
 Welcome at our website
 We hope you like it here
+And we hope you will stay for a while
```



# PRZYKŁAD STAGOWANIA (GIT ADD)

```
Terminal
We hope you like it here
And we hope you will stay for a while
tommy@Bielak ~/Projects/Website (master) $ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   index.html

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

tommy@Bielak ~/Projects/Website (master) $ git diff
diff --git a/index.html b/index.html
index ded25b6..fb6438e 100644
--- a/index.html
+++ b/index.html
@@ -1,2 +1,3 @@
 Welcome at our website
 We hope you like it here
+And we hope you will stay for a while
tommy@Bielak ~/Projects/Website (master) $ git diff --staged
diff --git a/index.html b/index.html
index 04dae32..ded25b6 100644
--- a/index.html
+++ b/index.html
@@ -1 +1,2 @@
 Welcome at our website
+We hope you like it here
tommy@Bielak ~/Projects/Website (master) $
```



Pytania?

# PRZYKŁADOWE FLOW

- Demo - pokazywanie statusu w SourceTree
- SourceTree - popularne, darmowe narzędzie.  
W czasie warsztatów czasem będzie wykorzystane do zaprezentowania efektów.

# ĆWICZENIE *PROSTE FLOW*

[git-presentation.tk/cwiczenia](http://git-presentation.tk/cwiczenia)

Demo + samodzielna praca

Pytania?

# 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?

# PRZEGŁĄDANIE HISTORII

- git log
- gitk

Git log oraz gitk domyślnie pokazują historię od *obecnego momentu* - HEAD, ale możliwe jest wskazanie dowolnego momentu w historii, np. nazwy brancha, SHA commita lub odniesienia HEAD<sup>^</sup> lub HEAD~N

# POPULARNE PRZEŁĄCZNIKI DO GIT LOG / GITK

- git log -4
- git log --oneline
- git log --graph
- git log --decorate
- git log --stat
- git log --patch
- git log --since=X,git log --after=X,git log --before=X
- git log X --not Y,git log Y..X,git log X..Y
- git log --author="xyz"
- git log --grep="xyz"
- git log --no-merges
- git log --all
- git log --pretty="FORMAT"
- git log -S"text"

# GIT LOG Z GRAFEM

```
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 Z JEDNOLINIWYMI WYNIKAMI

```
| * 53ba33d Fix android-P repo
| * d8b646b Remove flaky test
tommy@Bielak ~/Desktop/temp/material-components-android (master) $ git log -5 --oneline
392aecा 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
* 392aecा 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
* 392aecा (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 Z POKAZANĄ STATYSTYKĄ I DIFFEM

```
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-androi
d/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

material-components-android: All files - gitk

File Edit View Help

master Update readme file  
Merge branch 'travisci'  
travisci remotes/origin/travisci Add lynx back  
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  
Add findbugs den for tests

SHA1 ID: d8b646bb3842161879bbcf7a0f9231580dba4b9e

Find   commit containing:  Exact

Diff Old version New version Lines of context:

Author: Cameron Ketcham <cketcham@google.com> 2018-05-18 22:03:32  
Committer: Cameron Ketcham <cketcham@google.com> 2018-05-18 22:02:19  
Parent: [9dbf97998002b1b6ec13878ea5ddce1199f52e05](#) (Modif.)  
Child: [53ba33d800788116945eb06fd34f26e89f9292ad](#) (Fix)  
Branches: [master](#), [remotes/origin/travisci](#), [travisci](#)  
Follows: [1.0.0-alpha1](#)  
Precedes:  
  
Remove flaky test

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@nonale.com> 2018-05-17 17:52:56

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

# PRZYGOTOWANIE DO ĆWICZEŃ Z PRZEGŁĄDANIEM HISTORII

1. Przejdź do katalogu cwiczen na pulpicie:

```
cd ~/Desktop/cwiczenia
```

2. Sklonuj przykładowe repozytorium (z projektem open source), do katalogu log-demo:

```
git clone https://gitlab.com/terrakok/gitlab-client log-demo
```

3. Przejdź do katalogu repo: cd log-demo

4. Zresetuj repozytorium do wersji:

```
7ad14ecda2f97a3ff7c3cca44cc7605484b578b5
```

```
git reset --hard 7ad14
```

*Żeby każdy miał tę samą wersję do ćwiczeń*

# ĆWICZENIE 1

Sprawdź datę ostatniego commita

WIZUALNIE:

gitk

Z KONSOLI:

git log -1

Odpowiedź:

Fri Dec 14 00:39:27 2018 +0300

# ĆWICZENIE 2

Sprawdź datę przedostatniego commita

**WIZUALNIE:**

gitk

**Z KONSOLI:**

git log -2

lub

git log HEAD^ -1

Odpowiedź:

Thu Dec 13 23:58:11 2018 +0300

# ĆWICZENIE 3

Sprawdź tytuł ostatniego commita autora z nazwiskiem  
*Gulya*

**WIZUALNIE:**

`gitk --author=gulya`

**Z KONSOLI:**

`git log -1 --author=gulya`

**Odpowiedź:**

Upgrade Markwon to version 2.0.0. Replace ImageSizeResolver workaround with upstream one.

# ĆWICZENIE 4

Sprawdź autora commita zawierającego w opisie frazę

*Update Stub*

**WIZUALNIE:**

```
gitk --all --grep="Update Stub"
```

**Z KONSOLI:**

```
git log --all --grep="Update Stub"
```

Odpowiedź:  
terrakok

# ĆWICZENIE 5

Sprawdź jakie pliki modyfikował commit  
b9272a3be3d4f9182e6893a0f9a2f7b9ff0b6923

**WIZUALNIE:**

gitk b927

**Z KONSOLI:**

git log b927 -1 --stat

**Odpowiedź:**

.../model/repository/session/SessionRepository.kt | 9 ++++++++  
.../terrakok/gitlabclient/presentation/auth/AuthPresenter.kt | 12 ++++++-----

# ĆWICZENIE 6

Sprawdź dokładnie co zmienił commit

0e02008383a59f6ab56a14df4688070aab925765

**WIZUALNIE:**

gitk 0e020 -1

**Z KONSOLI:**

```
git log 0e020 -1 --patch  
lub  
git show 0e020
```

**Odpowiedź:**

**Linie dodane** do pliku PrivacyPolicyFragment.kt:

```
override val parentScopeName = DI.APP_SCOPE
```

posta

```
Toothpick.inject(this, scope)
```

**Linia usunięta** z pliku PrivacyPolicyFragment.kt:

```
Toothpick.inject(this, Toothpick.openScope(DI.APP_SCOPE))
```

# ĆWICZENIE 7

Sprawdź id commitów z drugiego grudnia 2018

**WIZUALNIE:**

```
gitk --after="2018-12-02 00:01" --before="2018-12-02 23:59"
```

**Z KONSOLI:**

```
git log --after="2018-12-02 00:01" --before="2018-12-02 23:59"
```

**Odpowiedź:**

```
6167018 Merge branch 'task/upgrade-markwon-and-remove-image-loading-crutch' into 'develop'  
fdb2e6c Merge branch 'project_milestone_tab' into 'develop'  
158bba8 Delete unused "todo".  
6d45888 Add account id for fixing same userId for difference servers.  
0e02008 Fix scope for PrivacyPolicyFragment.  
c46c806 Merge branch 'feture/multi_account' into develop  
45c0836 Delete code style from git history.  
5752be0 Add scopes to each fragment for more powerful memory management.
```

# ĆWICZENIE 8

Sprawdź kto podniósł wersję produktu z 9 na 10  
(text: `versionCode 10`)

Z KONSOLI:

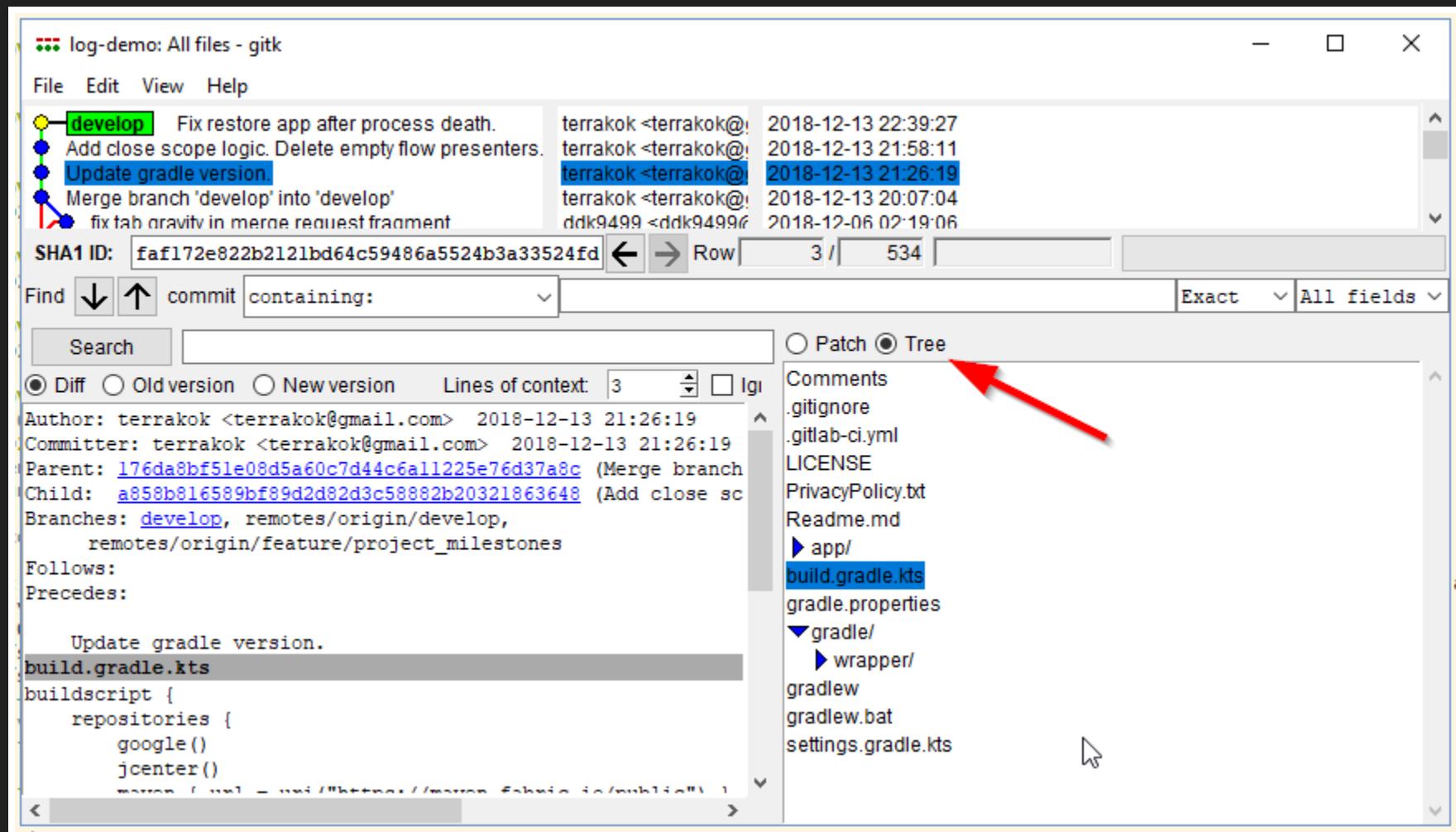
```
git log -S "versionCode 10" -p
```

Odpowiedź:

*terrakok, w commicie*

`6d289b10a07bb13d75a7b9767b9b0ae52b0421ce`

# GITK POKAZUJĄCY CAŁY STAN PROJEKTU



Przejrzeć drzewo można również w konsoli, ale to się przydaje głównie do skryptów  
git ls-tree -r 0e020 --name-only

Pytania?

# 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

# **ĆWICZENIE 'ZABAWA' Z RESETOWANIEM COMMITÓW I OBSERWACJA 'CO SIĘ DZIEJE'**

[git-presentation.tk/cwiczenia](http://git-presentation.tk/cwiczenia)

# EDYCJA 'OSTATNICH' COMMITÓW PO OPUBLIKOWANIU

`git revert`

np:

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

# DEMO REBASE --INTERACTIVE

```
pick 95a10ac New cars
pick 91b7878 Add wrong car
pick d294edd Add Mazdaaa car
pick 2973b7f Add Skoda
pick 2c053d1 Correct typo
pick 5ae3037 Add description

# Rebase ff03ffb..5ae3037 onto ff03ffb (6 commands)
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
```

# DEMO REBASE --INTERACTIVE

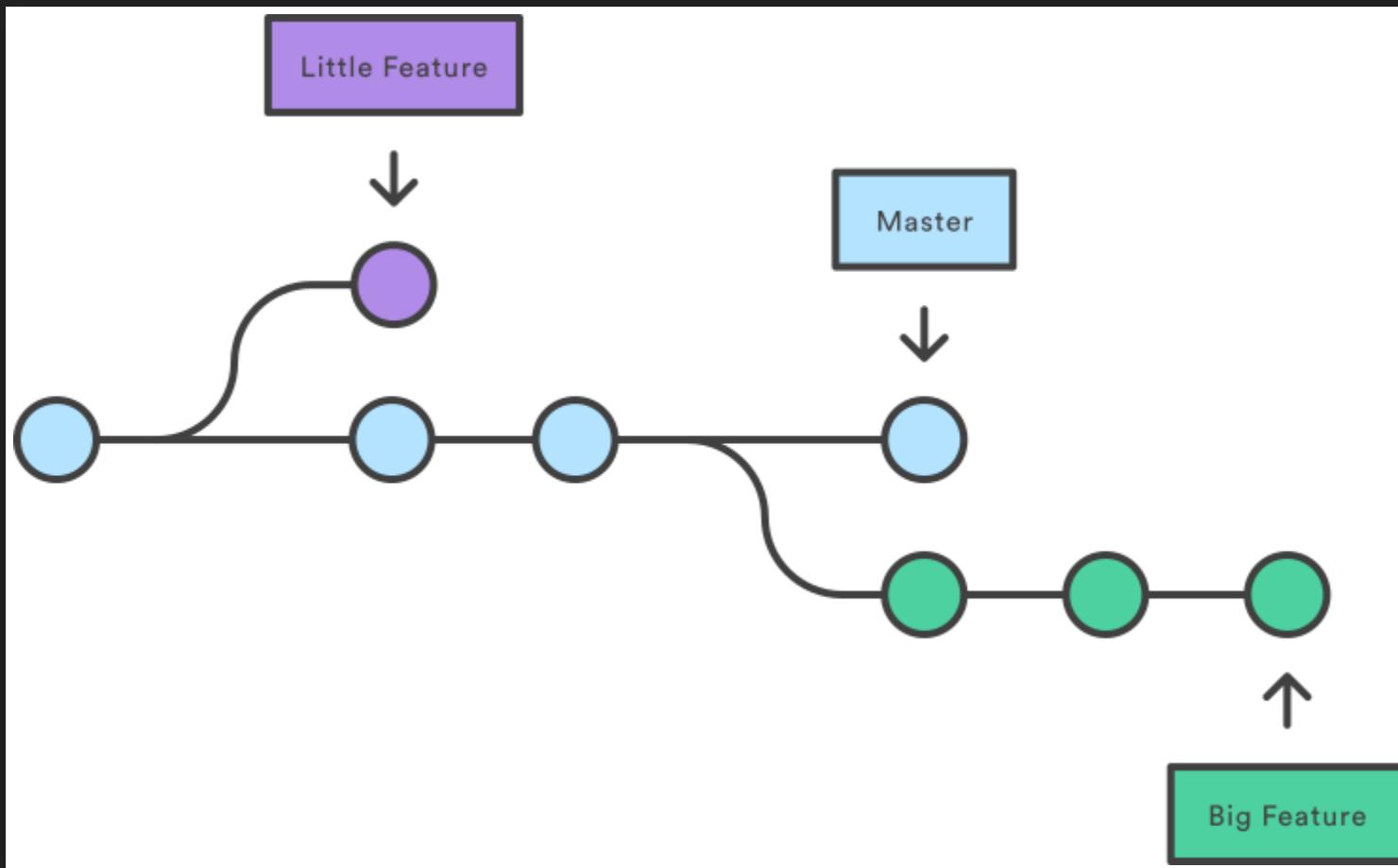
- git rebase -i HEAD~6
- git rebase --interactive HEAD~6
- git add .
- git commit --amend
- git rebase --continue
- Przenoszenie linii tekstu(hashe) w edytorze tekstu

# **ĆWICZENIE POPRAWIANIE COMMITÓW**

[git-presentation.tk/cwiczenia](http://git-presentation.tk/cwiczenia)

Pytania?

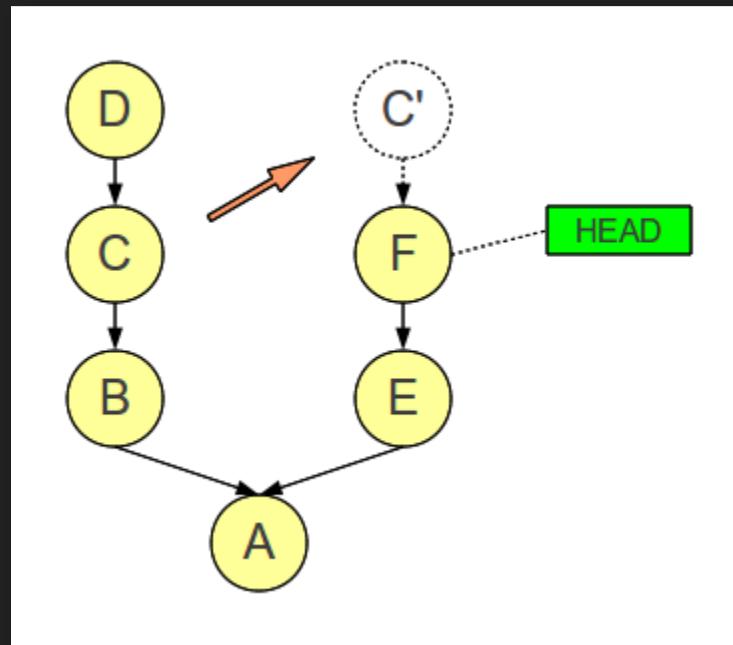
# BRANCHE



Learning branching

# CHERRY-PICK

Commity między branchami można przekładać



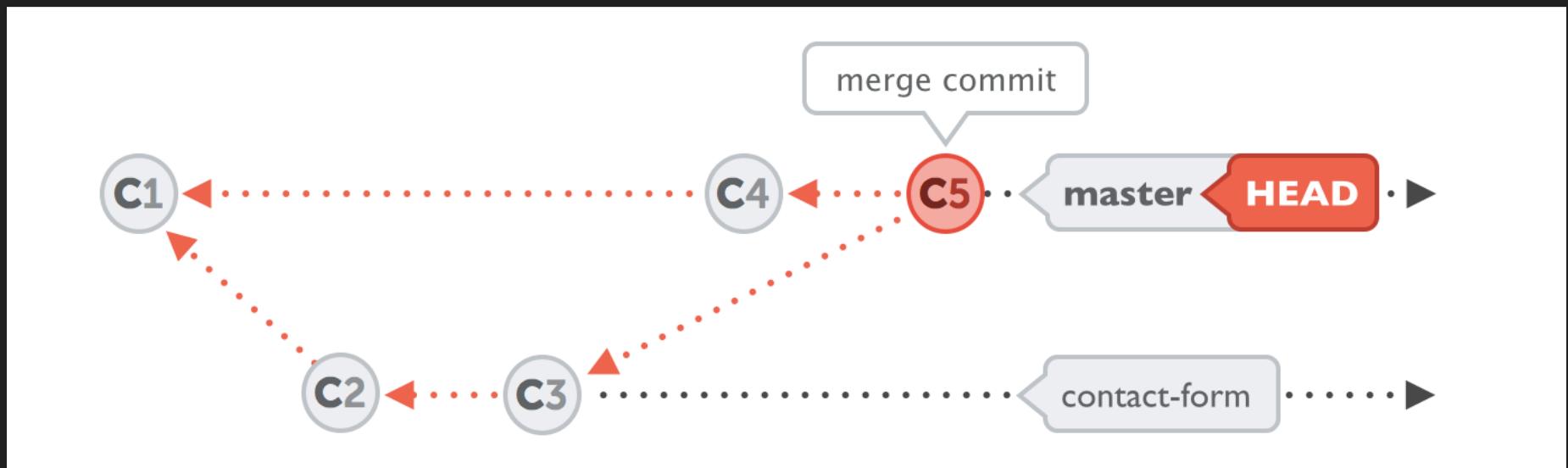
# DEMO

pokazanie jak zmiana brancha wpływa na zawartość  
katalogów (git checkout)

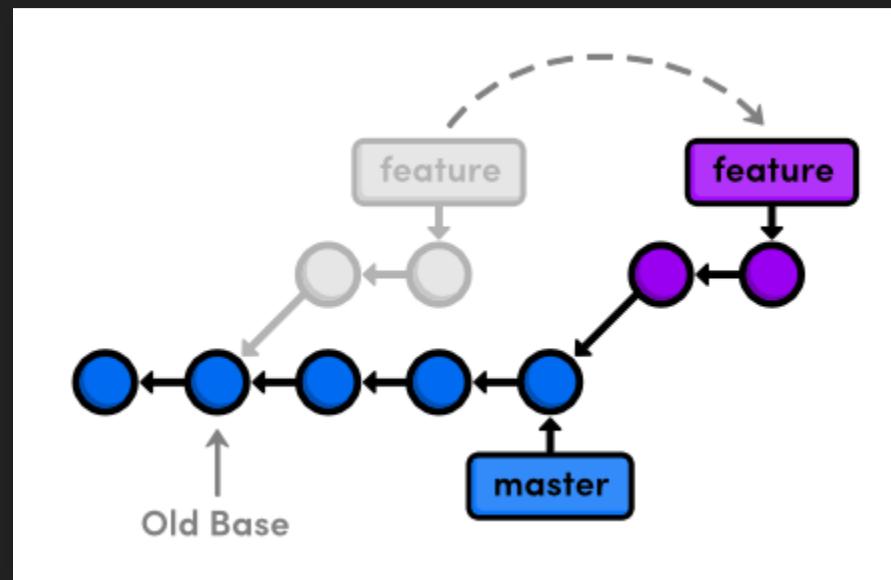
stosowanie cherry-pick w praktyce

Pytania?

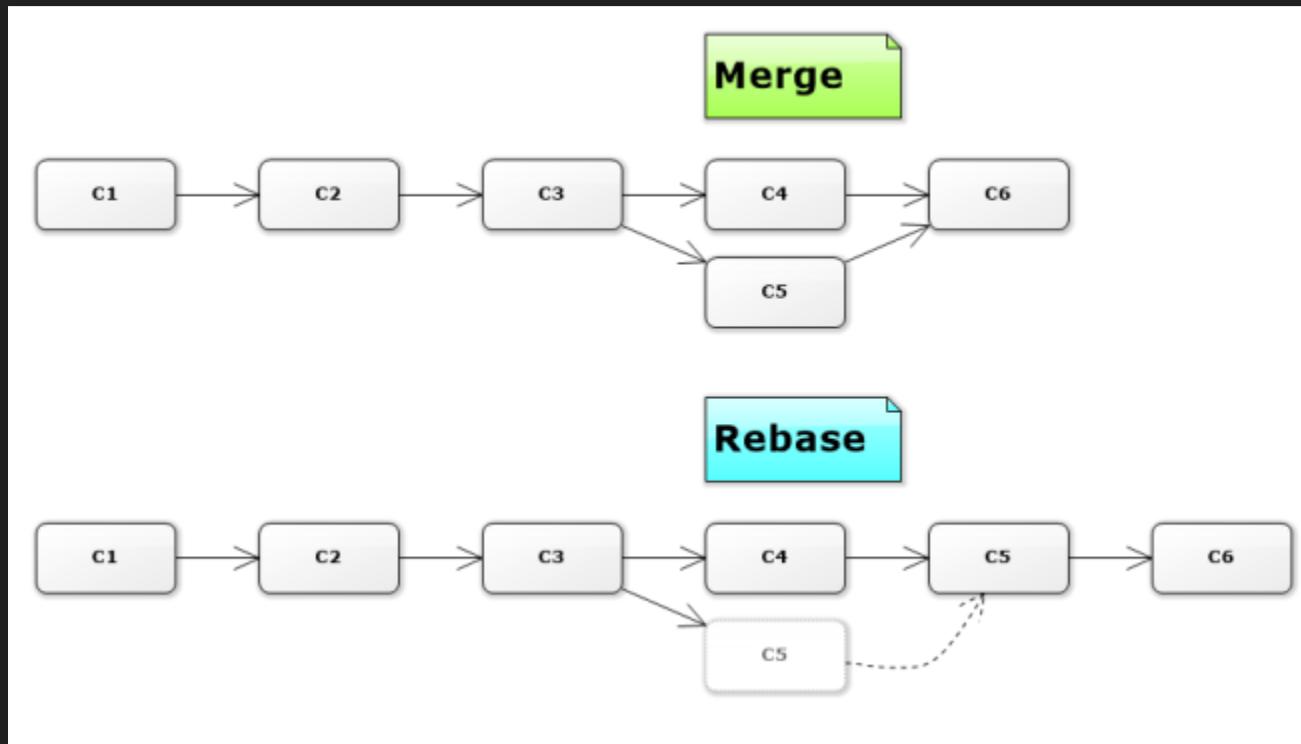
# MERGE



# REBASE



# MERGE VS REBASE



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

[git-presentation.tk/cwiczenia](http://git-presentation.tk/cwiczenia)

# REBASE TO ŚWIETNE NARZĘDZIE...

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

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

(Jeżeli flow projektu na to pozwala)

*link do fajnego artykułu tłumaczącego trudności które można napotkać*

*Trunk-Based Development or Pull Requests - Why Not Both?*

- rebase zakłamauje 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 dokładna dodatkowej pracy

Pytania?

# 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

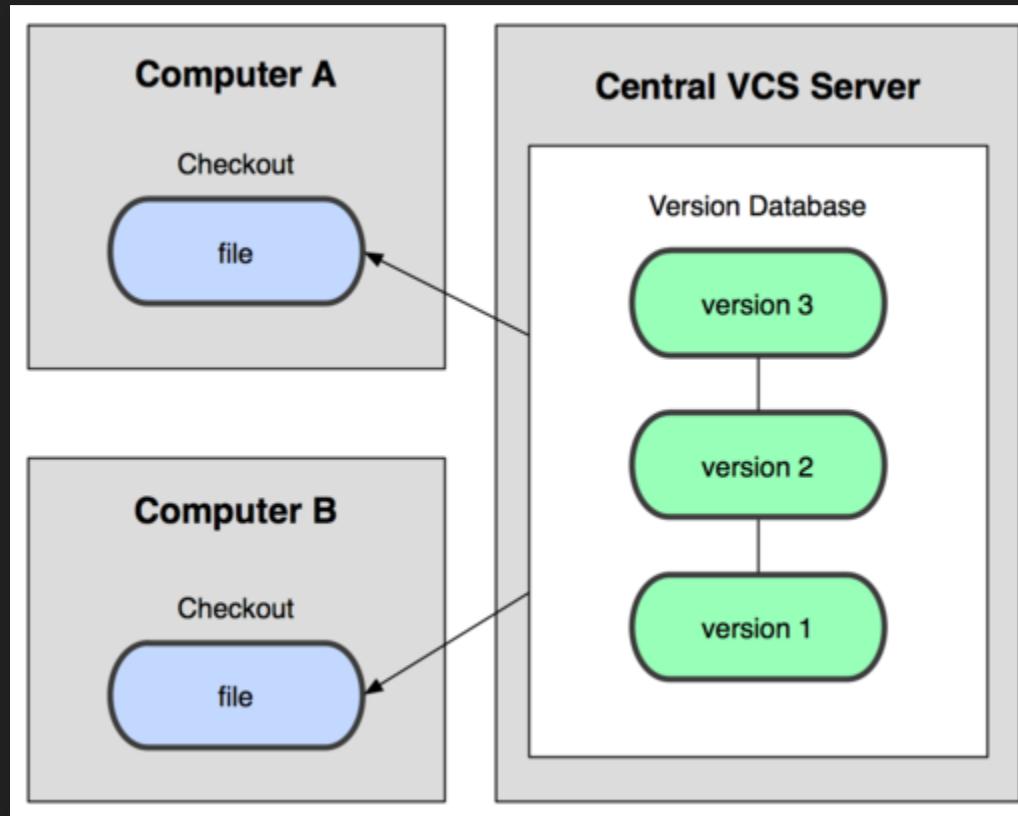
- Zawartości i nazwach wszystkich plików
- ID parent commit(ów)
- Wiadomości (opisie) commita
- Autorze i/lub commiterze

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

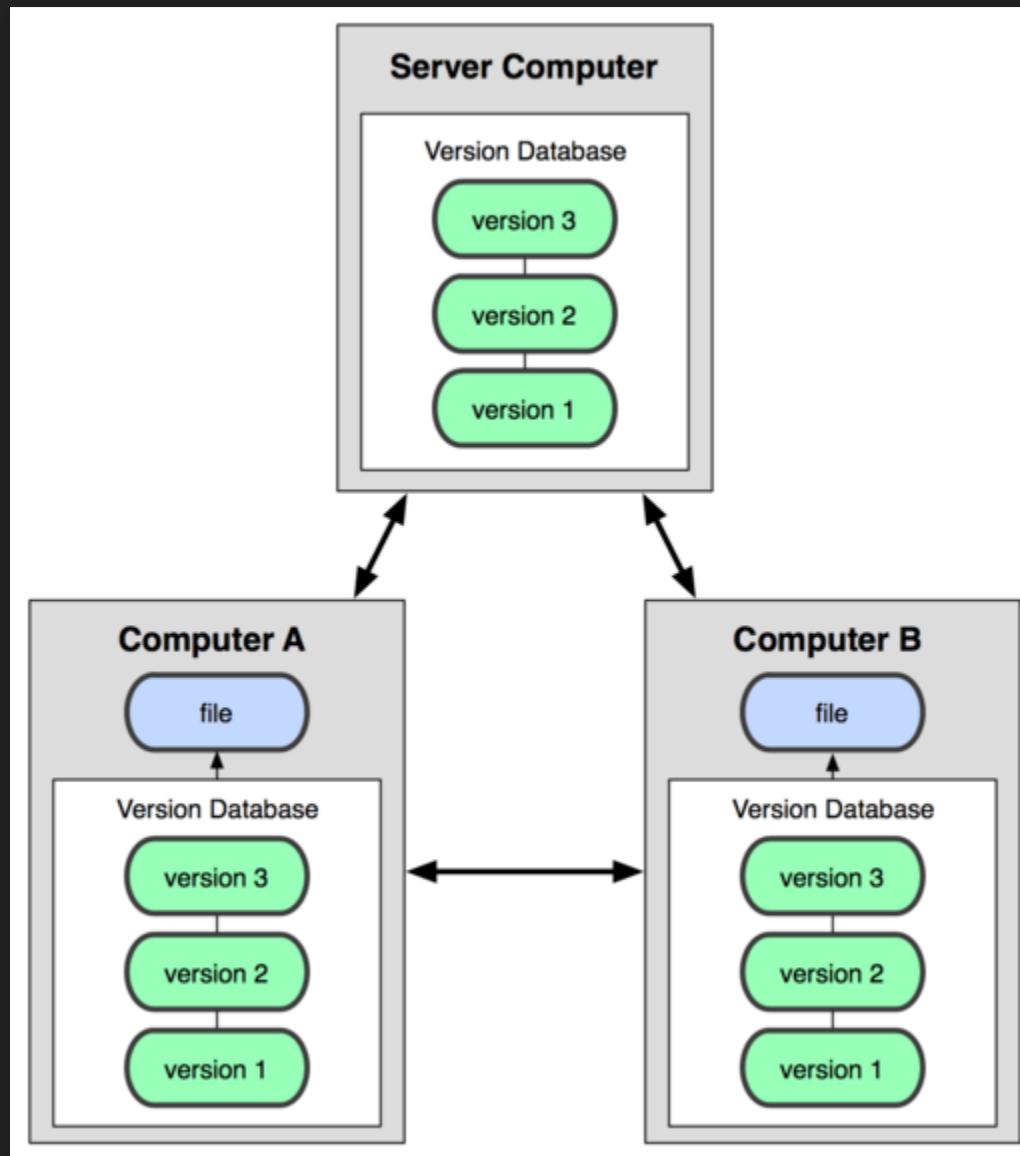
Analogicznie wiarygodne są przelewy w bitcoin

Pytania?

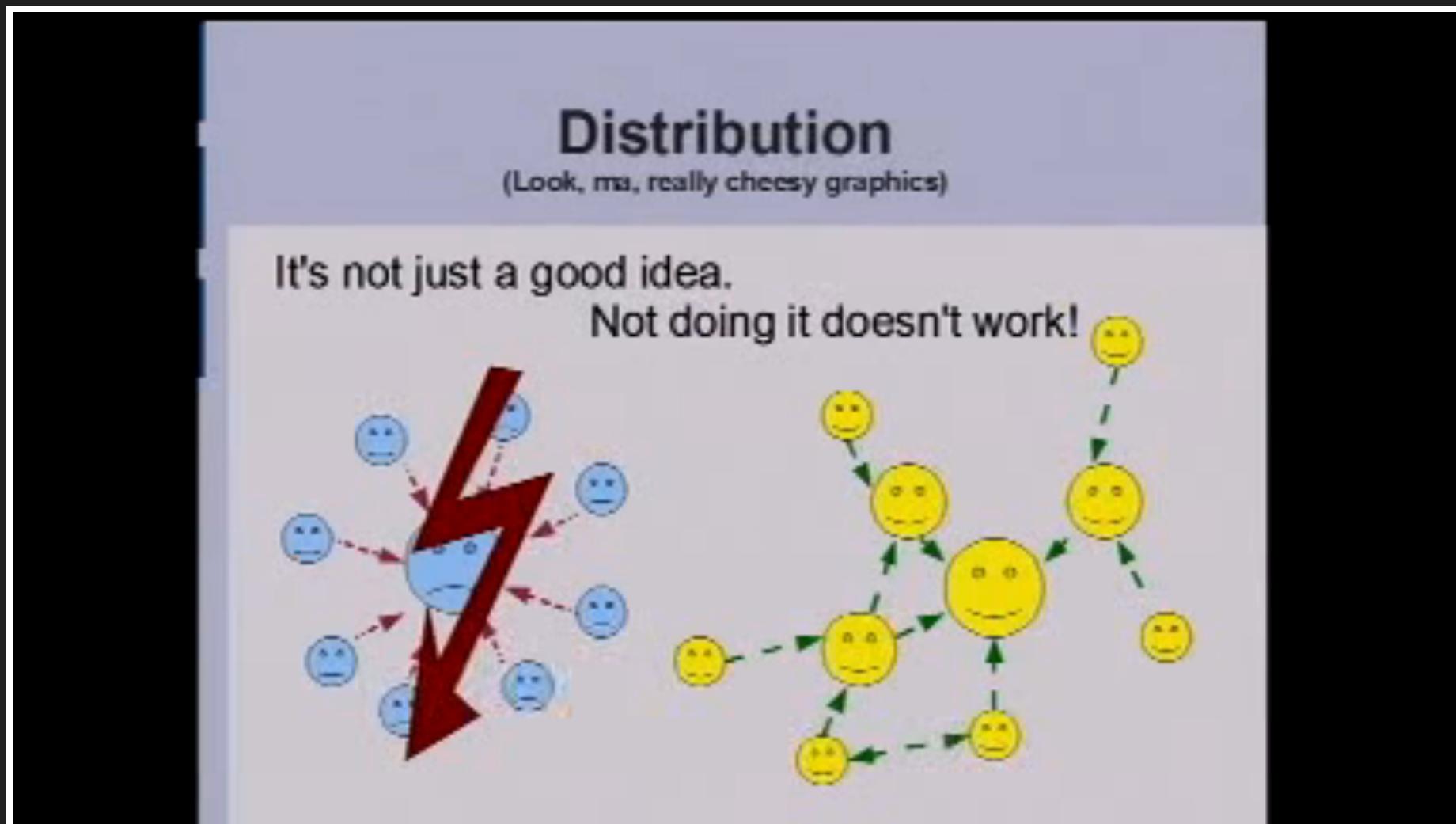
# Git nie jest scentralizowanym systemem kontroli wersji



# Git jest zdecentralizowanym systemem kontroli wersji

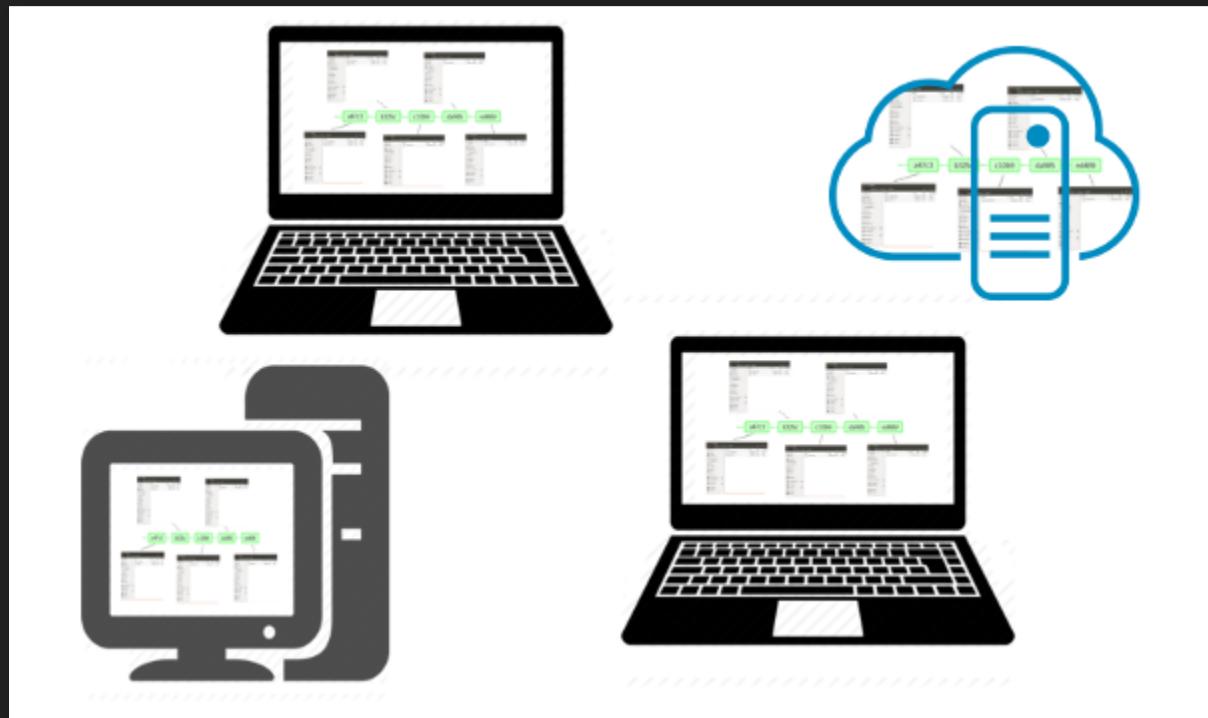


Git jest zdecentralizowanym systemem kontroli wersji



# GIT TO W ZASADZIE BAZA DANYCH

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



Pytania?

# 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
```

## PUSH --FORCE

Jeżeli jesteś pewny co robisz, możesz wymusić pusha

`git push --force`

Ale jak już wymuszasz, to zawsze z zabezpieczeniem

`git push --force-with-lease`

Pytania?

# IGNOROWANIE PLIKÓW

Ścieżki do plików/katalogów które chcemy ignorować dla repozytorium, trzymamy w pliku .gitignore

Plik ten wersjonujemy w repozytorium

# PRZYKŁADOWE REGUŁY IGNOROWANIA

- \*.orig
- \*\*/[Pp]ackages/\*
- bin/

## DODAJĘ REGUŁĘ DO .GITIGNORE, A "GIT STATUS" DALEJ POKAZUJE ZMIANY

Jest to scenariusz na który każdy przedzej czy później się natknie

Powód jest prosty - plik już jest w repozytorium,  
.gitignore ignoruje tylko nie śledzone pliki

Wystarczy plik... usunąć z repozytorium

# USUWANIE PLIKU Z REPOZYTORIUM, ŻEBY "POSŁUCHAŁ" REGUŁ .GITIGNORE

- Tradycyjnie:  
usunięcie pliku -> git add -> git commit
- Za pomocą komendy:  
`git rm file1.txt`  
`git commit -m "remove file1.txt"`
- Zostawiając plik na dysku:  
`git rm --cached file1.txt`  
`git commit -m "remove file1.txt"`

# GOTOWE REGUŁY .GITIGNORE

- [www.gitignore.io](http://www.gitignore.io)
- checkbox przy tworzeniu nowego projektu
- *mój ;) .gitignore*

Pytania?

# 7 ZASAD DOBREGO COMMIT MESSAGE

1. Oddziel tytuł od ciała pustą linią
2. Ogranicz tytuł do **50 znaków**
3. Stosuj wielkie litery w tytule
4. Nie kończ tytułu commita kropką
5. Zapisuj tytuł w trybie rozkazującym
6. Ogranicz 'szerokość' ciała do 72 znaków
7. W ciele opisz *co* i *dlaczego*, a nie *jak*

<https://chris.beams.io/posts/git-commit/>

Pytania?

# GIT REFGLOG

```
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
```

Pytania?

# SCHOWEK

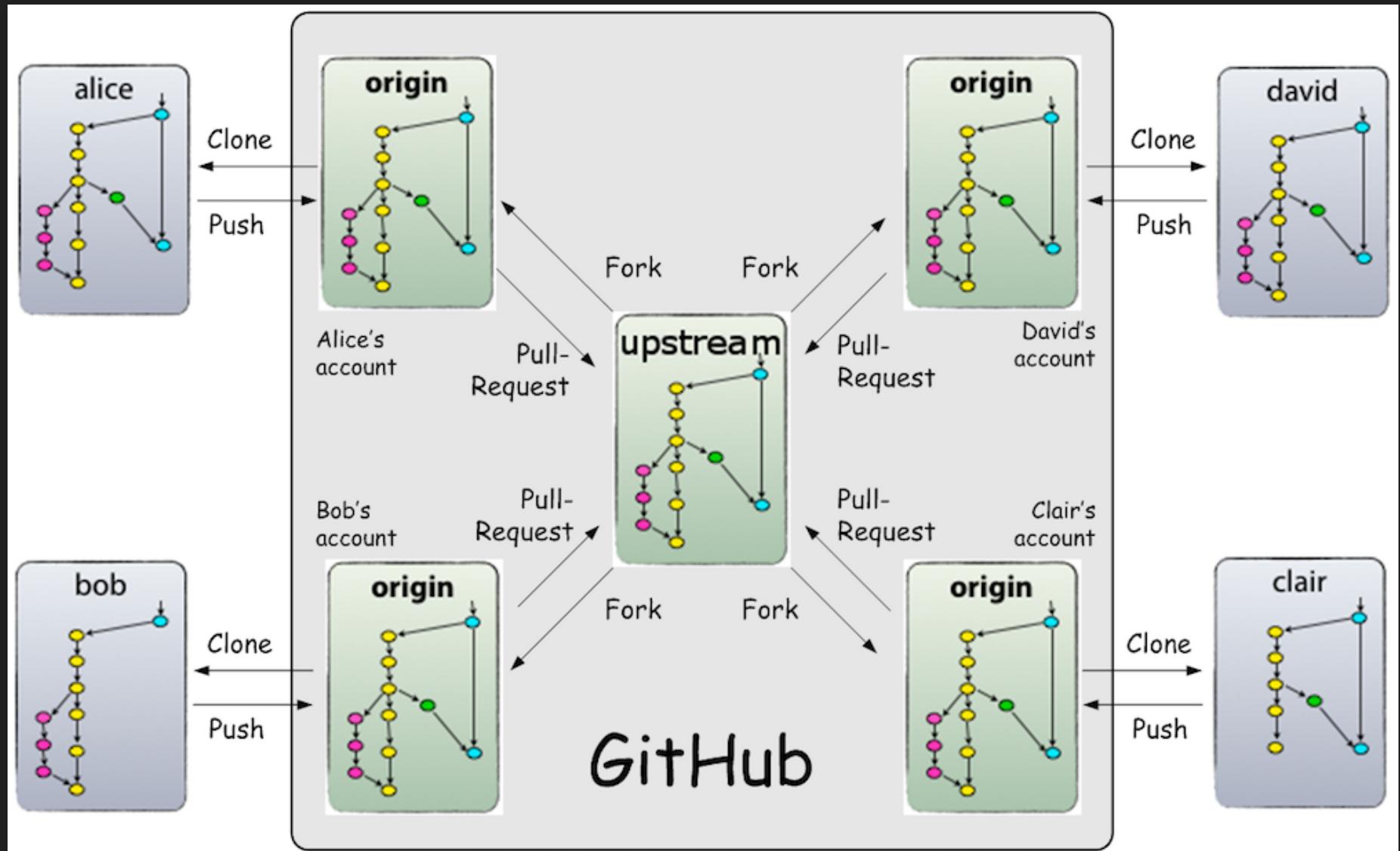
- git stash
- git stash --include-untracked
- git stash list
- git stash apply (optional: name)
- git stash pop (optional: name)
- git stash drop (optional: name)
- Różne wariacje, zachowywanie indexu, nadawanie wiadomości itd.

## Uwaga

Łatwo zapomnieć o nieśledzonych plikach

Pytania?

# FORKI & PULL REQUESTY

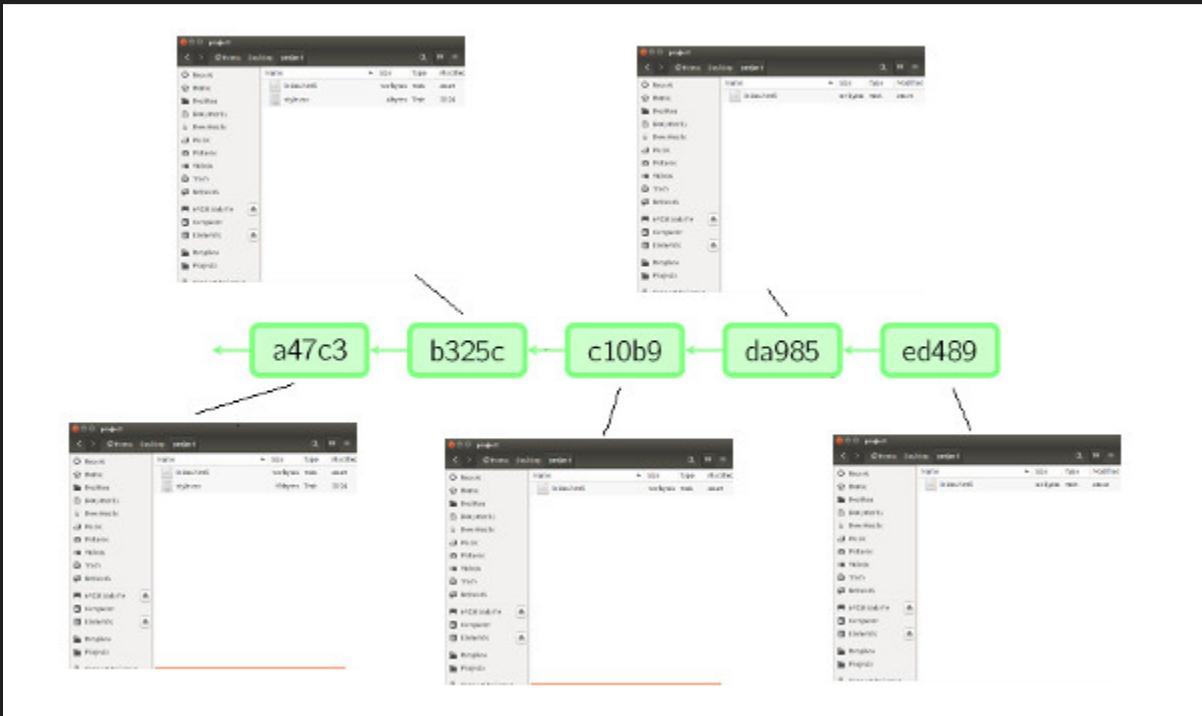


# PYTANIA? - PULL REQUESTY!

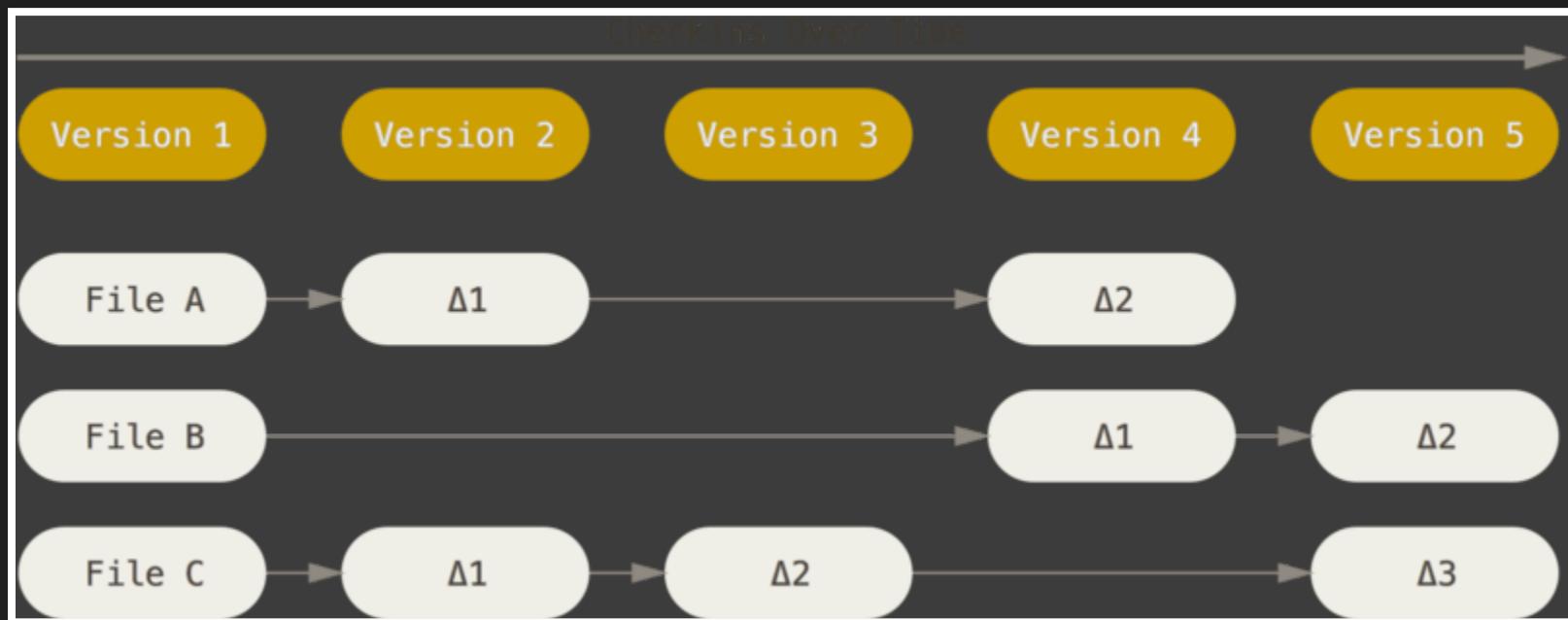
[git-presentation.tk/cwiczenia](http://git-presentation.tk/cwiczenia)

# JAK DZIAŁA GIT

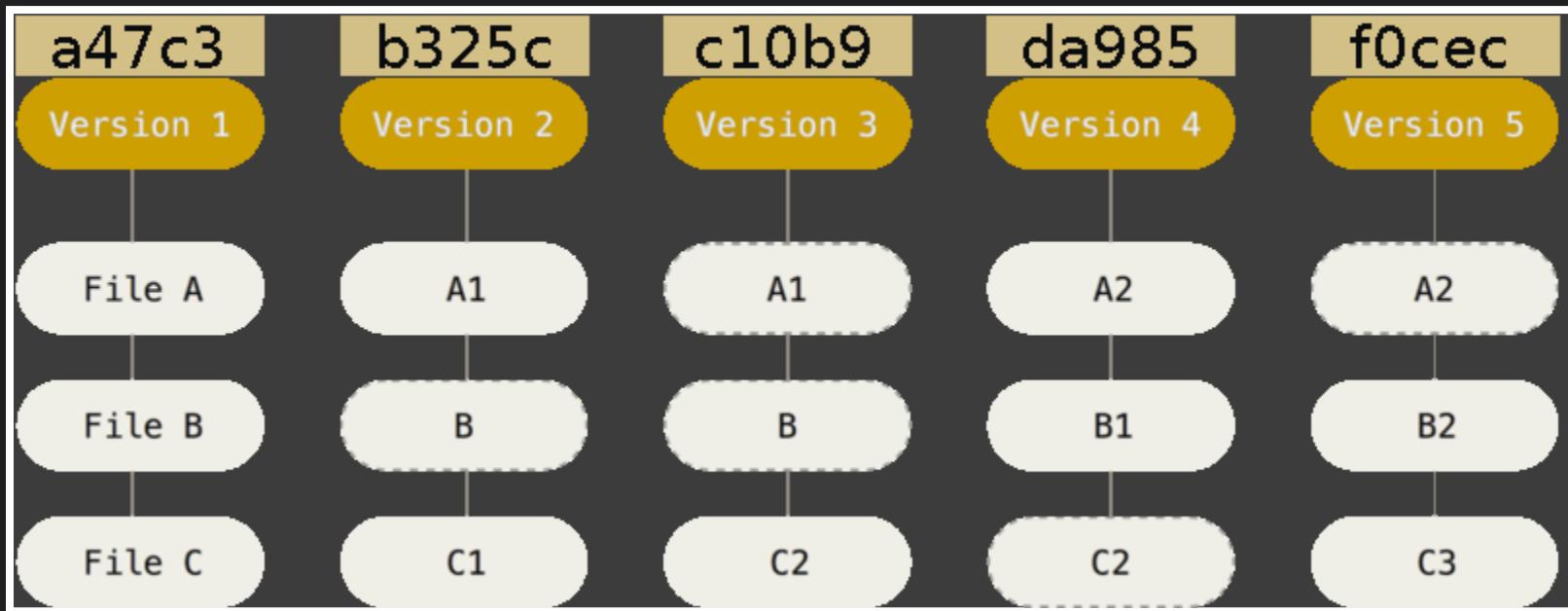
Git w podstawowym scenariuszu jedynie dodaje kolejne snapshoty plików



## 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 PRZECIWIEŃSTWIE 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

```
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, doszędł nowy  
plik, a jeden istniejący zniknął

Pytania?

# PRZYDATNE NARZĘDZIA ZWIĄZANE Z GITEM

- GitHub Pages / GitLab Pages / Netlify
  - Np. prezentacja na którą właśnie patrzymy:  
[git-presentation.tk](http://git-presentation.tk)
- GitHub gist

Pytania?

# GIT BLAME

Który to ?!

Terminal

```
+ tometchy@bielak ~/Projects/git-dev-warsztaty-presentation/slides (master) $ git blame -L3,15 index.html
x 05fddb10 (Tometchy 2018-05-13 19:16:27 +0200 3)
f64e2ff3 (Tometchy 2018-05-14 20:42:33 +0200 4) <head>
f64e2ff3 (Tometchy 2018-05-14 20:42:33 +0200 5)      <meta charset="utf-8">
05fddb10 (Tometchy 2018-05-13 19:16:27 +0200 6)
f64e2ff3 (Tometchy 2018-05-14 20:42:33 +0200 7)      <title>Git jest git</title>
05fddb10 (Tometchy 2018-05-13 19:16:27 +0200 8)
f64e2ff3 (Tometchy 2018-05-14 20:42:33 +0200 9)      <meta name="description" content="Wprowadzenie do git: ...
f64e2ff3 (Tometchy 2018-05-14 20:42:33 +0200 10)     <meta name="author" content="Krzysztof Morcinek, Toma...
05fddb10 (Tometchy 2018-05-13 19:16:27 +0200 11)
f64e2ff3 (Tometchy 2018-05-14 20:42:33 +0200 12)     <meta name="apple-mobile-web-app-capable" content="ye...
f64e2ff3 (Tometchy 2018-05-14 20:42:33 +0200 13)     <meta name="apple-mobile-web-app-status-bar-style" co...
05fddb10 (Tometchy 2018-05-13 19:16:27 +0200 14)
f64e2ff3 (Tometchy 2018-05-14 20:42:33 +0200 15)     <meta name="viewport" content="width=device-width, in...
```

Pytania?

# ALIASY

Modyfikacje aliasów można wykonywać w gitconfigu

```
[alias]  
st = status  
ci = commit -v
```

Albo za pomocą polecenia

```
git config --global alias.st status  
git config --global alias.ci 'commit -v'
```

Przykład użycia - skrócone wypisywanie statusu

```
git st
```

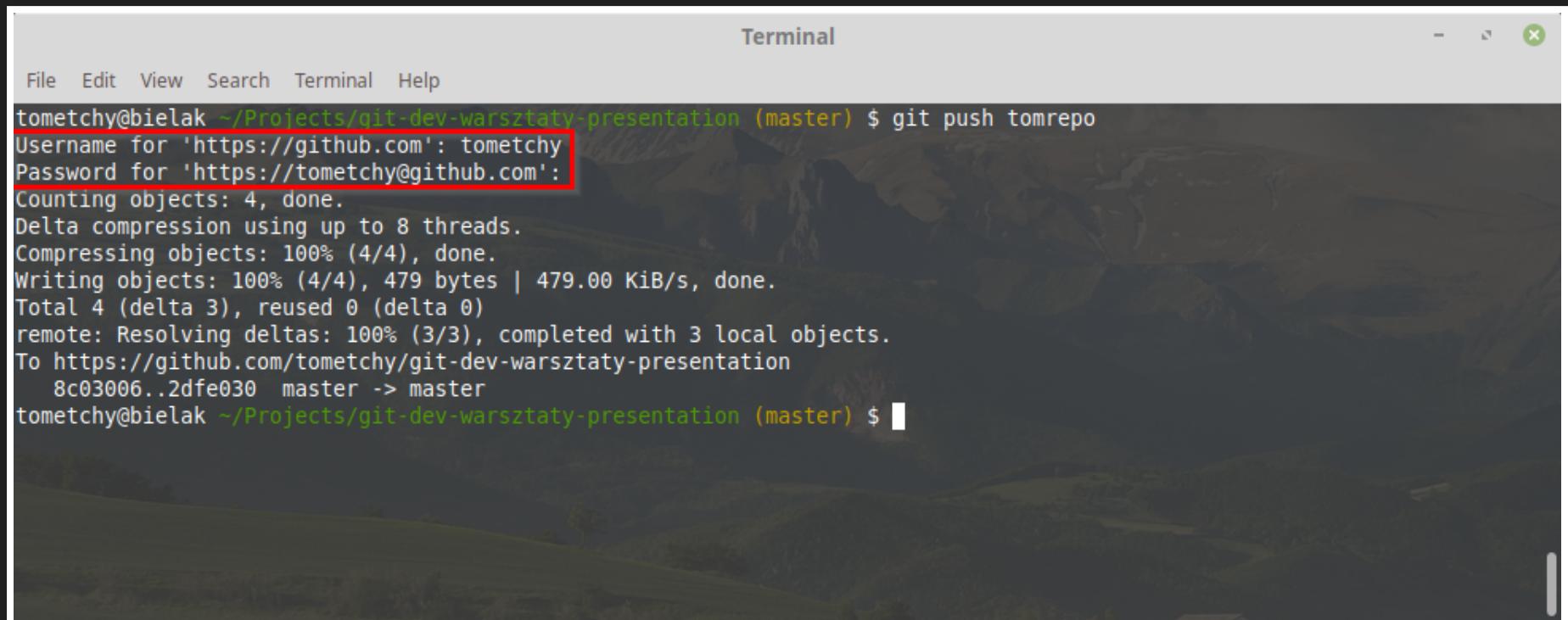
# INNE PRZYKŁADY ALIASÓW

```
wdiff = diff --word-diff=plain
rh1 = reset HEAD^ --hard
amend = commit --amend -aC HEAD
standup = log --since '1 day ago' --oneline
--author krzysztof.morcinek@gmail.com # hack
it with your email
cam = commit -am
jira = log --since '6am' --oneline --author
krzysztof.morcinek@gmail.com # hack it with
your email
ls = log --
pretty=format:"%C(yellow)%h%Cred%d\\\
%Creset%s%Cgreen\\ [%cn]" --decorate
mt = mergetool
```

Pytania?

# CREDENTIALS HELPER

Każdy push, fetch (czyli również pull), wymaga autoryzacji...



```
Terminal
File Edit View Search Terminal Help
tometchy@bielak ~/Projects/git-dev-warsztaty-presentation (master) $ git push tomrepo
Username for 'https://github.com': tometchy
Password for 'https://tometchy@github.com':
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 479 bytes | 479.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/tometchy/git-dev-warsztaty-presentation
  8c03006..2dfe030 master -> master
tometchy@bielak ~/Projects/git-dev-warsztaty-presentation (master) $
```

# WBUDOWANE CREDENTIALS HELPERY

- **cache**

```
git config --global credential.helper 'cache --timeout=300'  
Bezpieczny, ale tylko tymczasowy (in memory).  
Opcjonalnie własny timeout w sekundach (domyślny 900 - 15minut).
```

- **store**

```
git config --global credential.helper store  
Wygodny, ale niebezpieczny - credentiale zapisane w pliku tekstowym niezaszyfrowane.
```

# **LIBSECRET - WYGODNY I BEZPIECZNY CREDENTIALS HELPER DLA LINUX**

**Przykładowa instalacja dla Linux Mint / Ubuntu**

1. sudo apt-get install libsecret-1-0 libsecret-1-dev
2. cd /usr/share/doc/git/contrib/credential/libsecret
3. sudo make

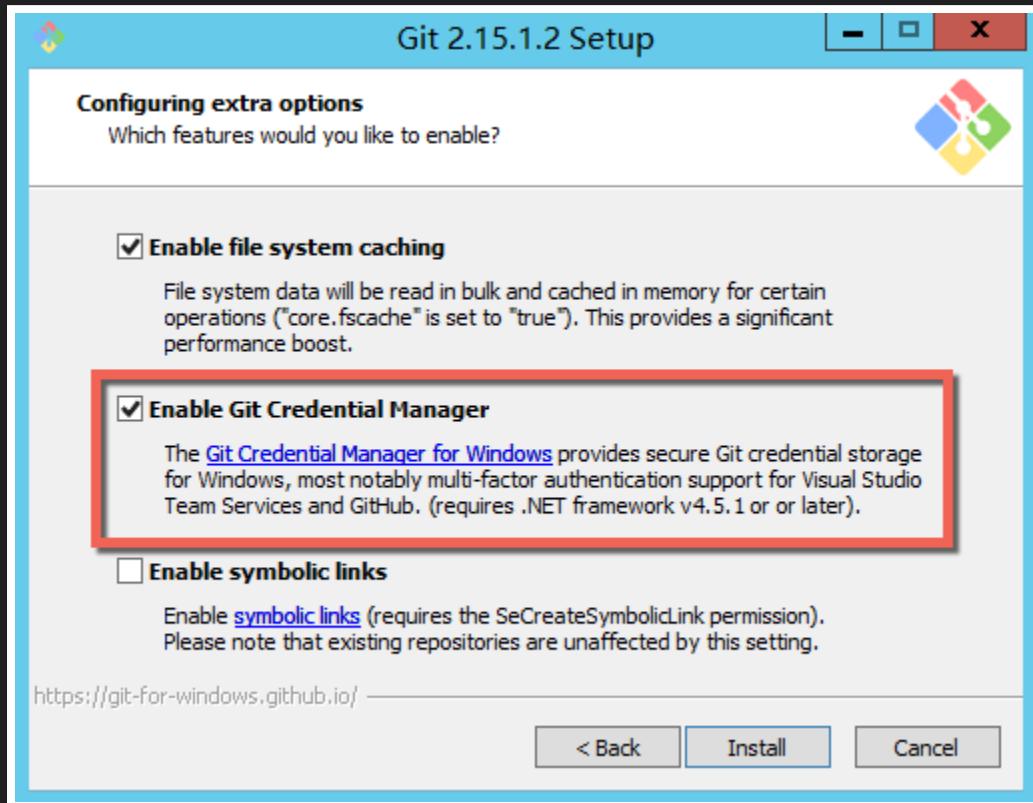
**Konfiguracja (gdy już zainstalowany)**

```
git config --global credential.helper  
/usr/share/doc/git/contrib/credential/libsecret/git-credential-libsecret
```

# WYGODNY I BEZPIECZNY CREDENTIALS HELPER DLA WINDOWS

Instalator: [github.com/Microsoft/Git-Credential-Manager-for-Windows/releases](https://github.com/Microsoft/Git-Credential-Manager-for-Windows/releases)

Lub *checkbox* w trakcie instalacji gita:



Powinno ustawić: `git config --global credential.helper manager`  
albo ścieżkę do exe

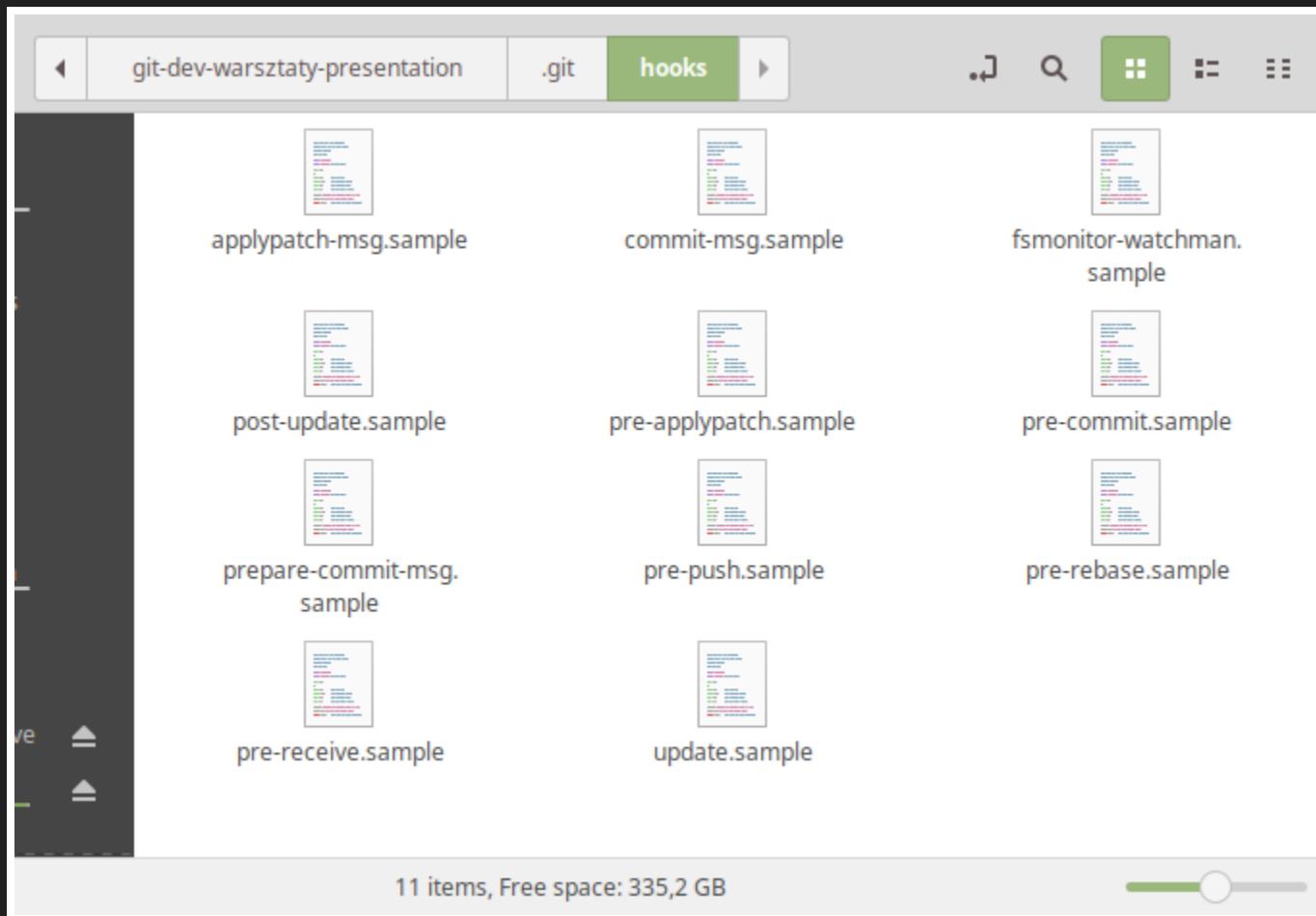
Pytania?

# PODPIS KRYPTOGRAFICZNY

Commity i tagi można podpisywać kryptograficznie, jednak wymaganie tego w workflow jest kłopotliwe

```
$ git log --show-signature
commit 5c3386cf54bba0a33a32da706aa52bc0155503c2
gpg: Signature made Wed Jun 4 19:49:17 2014 PDT using RSA key ID 0A46826A
gpg: Good signature from "Scott Chacon (Git signing key) <schacon@gmail.com>"
Author: Scott Chacon <schacon@gmail.com>
Date: Wed Jun 4 19:49:17 2014 -0700
Add new function
```

# GIT HOOKS



# LINKI

- Pro Git book (Scott Chacon, Ben Straub)
- How to Write a Git Commit Message  
(The seven rules of a great Git commit message)
- 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
- GIT Illustrated Cheatsheet
- Artykuł na co uważać przy rebase
- Trunk Based Development: Introduction

# 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
```