

# Version control systems

---

czyli co i jak z tymi gitami.

Opracowali: Paweł Jan Tłusty  
Viktor Vodnev  
e-mail: [hotor22@gmail.com](mailto:hotor22@gmail.com)



# Co to jest system kontroli wersji?

## In short VCS

---

Version control system to środowisko przechowujące wszystkie zmiany dokonane w projekcie zaprojektowany w sposób, aby równoległe mogło z niego korzystać wiele osób.

Może odpowiedzieć developerom na pytania typu:

- Jakie zmiany wprowadzono?
- Kto wprowadził zmianę?
- Kiedy ta zmiana została wprowadzona?

# Repozytorium i te sprawy

---

Git umożliwia nam do spojrzenia w pełną historię zmian repozytorium. Umożliwiając przy tym bezstratną możliwość przywracania poszczególnych części kodu.

>  **.git/**



Added



Changed



Renamed



Removed



# Ale co dla nas znaczy repozytorium

---

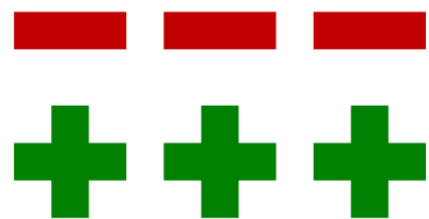
Na początek powinno nam wystarczyć, że jest to taka czarna skrzynka, która ma w środku jakąś strukturę, mechanizmy. Dla nas aktualnie liczy się to, że coś wkładamy do tego pudełka i oczekujemy konkretnego wyniku. Jest to literalnie idea programistycznej czarnej skrzynki



# Odrobina historii



Tux- maskotka systemu Linux

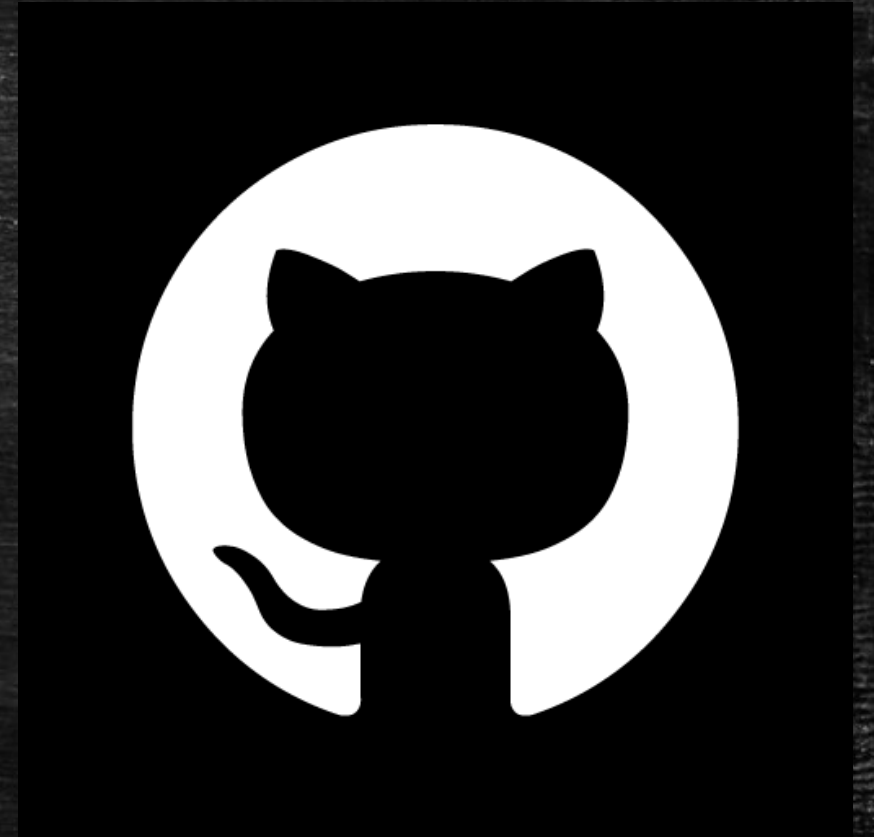


Linus Torvalds





!=



Git i GitHub to nie to samo. Git to zaawansowany system kontroli wersji najczęściej utrzymywany lokalnie. Zaś GitHub to serwis chmurowy umożliwiający hostowanie repozytoriów Gita.

# Podstawowe polecenia

- Tutaj wstawimy screena, z flagami -a -am rebase

## git add

- Przygotuj pliki
- Dodaj pliki do śledzenia

## git commit

- Zapisz „migawkę”
- Dokonaj wpisu do historii projektu

## git push

- Prześlij lokalne zmiany do repozytorium

```
pi@BackupComb:/media/pi/LINUX MINT/Semestr4 $ git add .
pi@BackupComb:/media/pi/LINUX MINT/Semestr4 $ git commit . -m'nazwa'
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
pi@BackupComb:/media/pi/LINUX MINT/Semestr4 $ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
pi@BackupComb:/media/pi/LINUX MINT/Semestr4 $ git push
Everything up-to-date
pi@BackupComb:/media/pi/LINUX MINT/Semestr4 $
```





# Podstawowe polecenia- przykład z GitHub


---

## Terminal



 Tworzymy zdalne repozytorium


 Inicjalizujemy repozytorium


 Łączymy je z serwisem

## GUI



 Tworzymy zdalne repozytorium

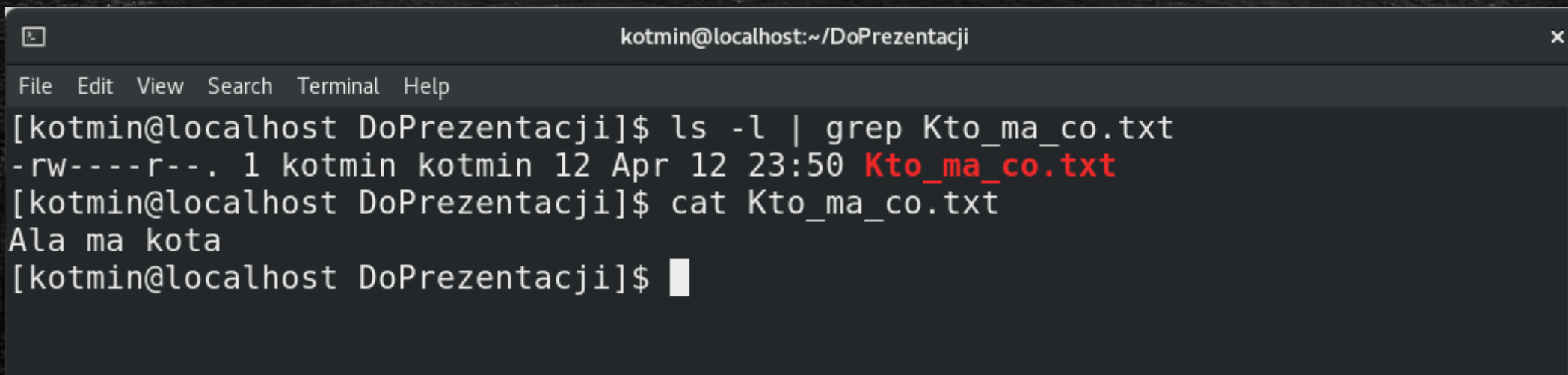
 Klonujemy istniejące repozytorium

 Nadpisujemy



# Ćwiczenie

Spróbujcie samodzielnie stworzyć lokalne repozytorium. Udostępnić je na waszym profilu w serwisie GitHub. Wysłać na nie plik „Kto\_ma\_co.txt” o następujących właściwościach:

A screenshot of a terminal window titled 'kotmin@localhost:~/DoPrezentacji'. The terminal shows the execution of two commands. The first command is 'ls -l | grep Kto\_ma\_co.txt', which outputs a line showing file permissions '-rw----r--', owner 'kotmin', group 'kotmin', date '12 Apr 12 23:50', and filename 'Kto\_ma\_co.txt'. The second command is 'cat Kto\_ma\_co.txt', which outputs the text 'Ala ma kota'.

```
kotmin@localhost:~/DoPrezentacji
File Edit View Search Terminal Help
[kotmin@localhost DoPrezentacji]$ ls -l | grep Kto_ma_co.txt
-rw----r--. 1 kotmin kotmin 12 Apr 12 23:50 Kto_ma_co.txt
[kotmin@localhost DoPrezentacji]$ cat Kto_ma_co.txt
Ala ma kota
[kotmin@localhost DoPrezentacji]$
```

Wskazówki:

- Do zainicjowania repozytorium służy polecenie `git init`
- Uprawnienia pliku możemy zmienić używając polecenia `chmod`
- Nie wszystko od razu musi działać, nie bójcie się sięgnąć po wyszukiwarkę

**Psst. Czytajcie uważnie komunikaty**

# Jak to można uprościć?

```
#!/bin/bash

cd /media/pi/LINUX' 'MINT/Semestr4
git pull
git add -A
git commit -am"`date`"
git push
echo 'Zakonczylem proces update dla Semestr4'
```

```
pi@BackupComb:~/Scripts $ ls -la | grep upd
-rwxr-xr-x  1 pi pi  147 Mar 10 13:08 update_repo.sh
```

```
pi@BackupComb:~/Scripts $ ./update_repo.sh
```

```
"update_repo.sh" 9 lines, 147 bytes
```




# Błędy, błędy, błędy

---

```
To github.com:Kotmin/Semestr4.git
! [rejected]          main -> main (non-fast-forward)
error: failed to push some refs to 'github.com:Kotmin/Semestr4.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
Zakonczylem proces update dla Semestr4
pi@BackupComb:~/Scripts $ git pull
fatal: not a git repository (or any of the parent directories): .git
pi@BackupComb:~/Scripts $
```

# Ale czy czegoś tutaj nam nie brakuje?

	Kotmin Swap table	344fb28 7 hours ago	11 commits
📁	CSS	137 line lucida grande	yesterday
📁	nbproject	Copy of 5_6_repo_without_css	2 days ago
📁	src	Copy of 5_6_repo_without_css	2 days ago
📄	README.md	Initial commit	2 days ago
📄	formularze.html	Copy of 5_6_repo_without_css	2 days ago
📄	galeria.html	Changeing class of article	yesterday
📄	index.html	Copy of 5_6_repo_without_css	
📄	tabele.html	Swap table	
📄	zamowienie.html	Copy of 5_6_repo_without_css	

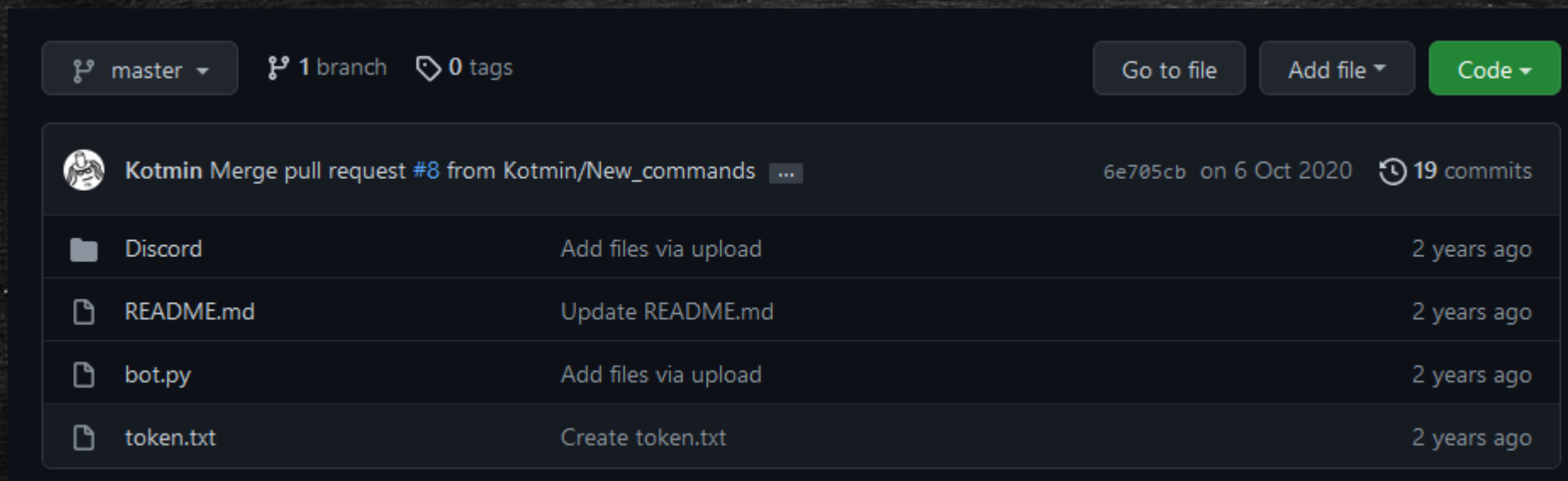
```
kotmin@localhost:~  
File Edit View Search Terminal Help  
[kotmin@localhost ~]$ ls -la | grep Kotmin.github.io  
drwxrwxr-x. 6 kotmin kotmin 177 Apr 24 14:25 Kotmin.github.io  
[kotmin@localhost ~]$ ls -l Kotmin.github.io/  
total 32  
drwxrwxr-x. 2 kotmin kotmin 51 Apr 24 14:25 CSS  
-rw-rw-r--. 1 kotmin kotmin 4132 Apr 24 14:25 formularze.html  
-rw-rw-r--. 1 kotmin kotmin 4228 Apr 24 14:25 galeria.html  
-rw-rw-r--. 1 kotmin kotmin 2473 Apr 24 14:25 index.html  
drwxrwxr-x. 3 kotmin kotmin 66 Apr 24 14:25 nbproject  
-rw-rw-r--. 1 kotmin kotmin 57 Apr 24 14:25 README.md  
drwxrwxr-x. 4 kotmin kotmin 54 Apr 24 14:25 src  
-rw-rw-r--. 1 kotmin kotmin 3246 Apr 24 14:25 tabele.html  
-rw-rw-r--. 1 kotmin kotmin 2195 Apr 24 14:25 zamowienie.html  
[kotmin@localhost ~]$
```



## Odsłóńmy mały kawałek tajemnicy

```
[kotmin@localhost ~]$ ls -la Kotmin.github.io/
total 36
drwxrwxr-x.  6 kotmin kotmin  177 Apr 24 14:25 .
drwx----- 26 kotmin kotmin 4096 Apr 24 14:25 ..
drwxrwxr-x.  2 kotmin kotmin   51 Apr 24 14:25 CSS
-rw-rw-r--.  1 kotmin kotmin 4132 Apr 24 14:25 formularze.html
-rw-rw-r--.  1 kotmin kotmin 4228 Apr 24 14:25 galeria.html
drwxrwxr-x.  8 kotmin kotmin  163 Apr 24 14:25 .git
-rw-rw-r--.  1 kotmin kotmin 2473 Apr 24 14:25 index.html
drwxrwxr-x.  3 kotmin kotmin   66 Apr 24 14:25 nbproject
-rw-rw-r--.  1 kotmin kotmin   57 Apr 24 14:25 README.md
drwxrwxr-x.  4 kotmin kotmin   54 Apr 24 14:25 src
-rw-rw-r--.  1 kotmin kotmin 3246 Apr 24 14:25 tabele.html
-rw-rw-r--.  1 kotmin kotmin 2195 Apr 24 14:25 zamowienie.html
```

# Oj czy aby na pewno wszystko powinno być publiczne?



Czy aby na pewno chcielibyśmy żeby coś takiego jak token wisiało na serwerze?



# .gitignore

---

jest to plik tekstowy „mówiący” systemowi kontroli wersji Git, które pliki ma ignorować w projekcie.

```
[kotmin@localhost Kotmin.github.io]$ cat .gitignore
# Prerequisites
*.d

# Compiled Object files
*.slo
*.lo
*.o
*.obj

# Precompiled Headers
*.gch
*.pch

# Compiled Dynamic libraries
*.so
*.dylib
*.dll

# Fortran module files
*.mod
*.smod

# Compiled Static libraries
*.lai
*.la
*.a
*.lib

# Executables
*.exe
*.out
*.app
```

# Jak samodzielnie się w tym odnaleźć?

---

`jakisplik.txt`

- Reguła dopasowania do pełnej nazwy
- Ominie tylko pliki o DOKŁADNIE takiej samej nazwie(A!=a)

`*.txt`

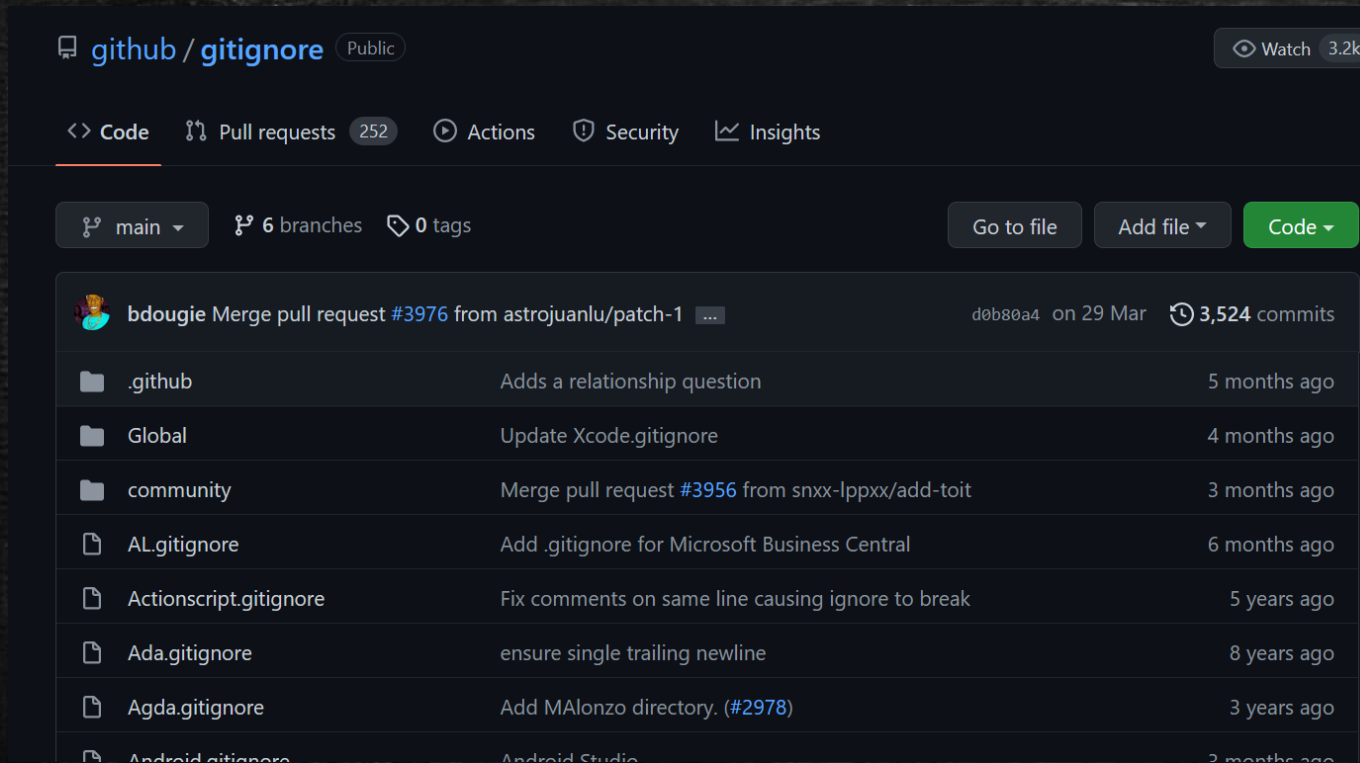
- \* Oznacza 0 lub wiele znaków
- Taka reguła zadziała dla plików z rozszerzeniem txt

`jakisplik[1-9].txt`

- Zadziała dla każdego pliku w formacie `jakisplik1.txt` ... `jakisplik9.txt`

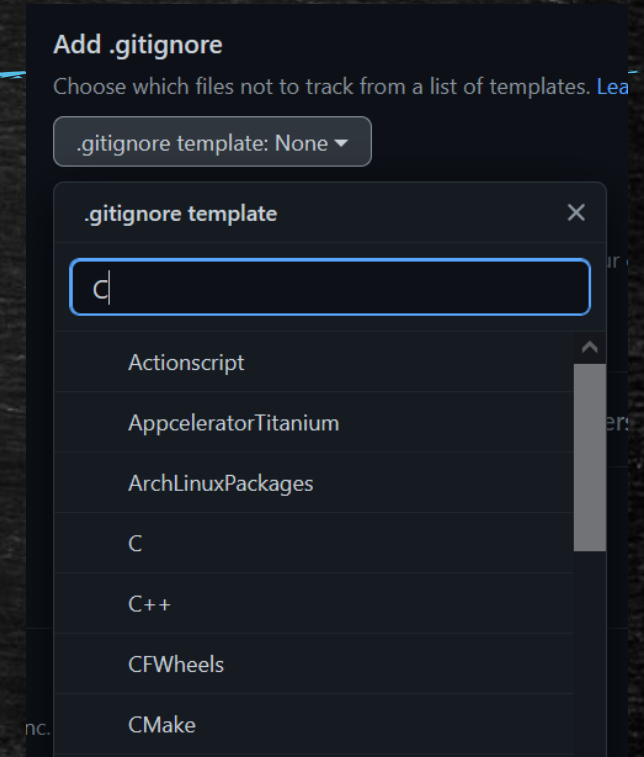


# I czy w ogóle trzeba to samemu pisać?



<https://github.com/github/gitignore>

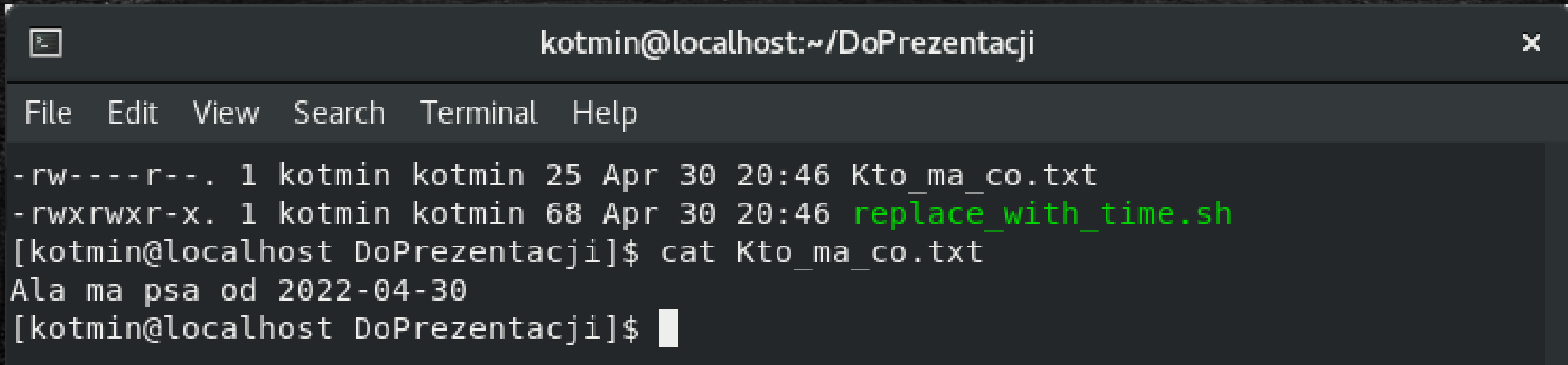
Projekt, w którym ludzie pro publico bono tworzą bazę danych plików .gitignore



Gotowe szablony w karcie tworzenia repozytorium na GitHub

# Ćwiczenie

Napiszcie skrypt wykonywalny (.sh) zmieniający zawartość pliku Kto\_ma\_co.txt na „Ala ma psa od”+(aktualna data w dowolnym formacie) oraz dodajcie regułę do pliku .gitignore rozkazującą ignorowanie plików z rozszerzeniem .sh:

A terminal window titled 'kotmin@localhost:~/DoPrezentacji' with a standard menu bar (File, Edit, View, Search, Terminal, Help). The terminal output shows the permissions and details for two files: 'Kto\_ma\_co.txt' and 'replace\_with\_time.sh'. The content of 'Kto\_ma\_co.txt' is displayed as 'Ala ma psa od 2022-04-30'.

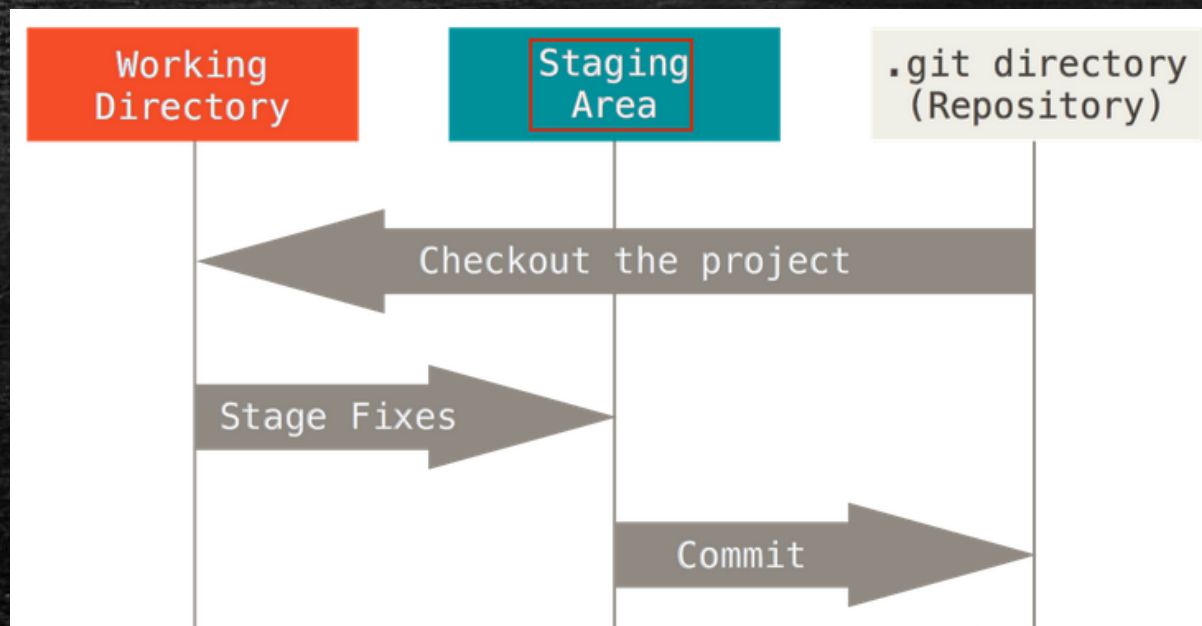
```
kotmin@localhost:~/DoPrezentacji
File Edit View Search Terminal Help
-rw----r--. 1 kotmin kotmin 25 Apr 30 20:46 Kto_ma_co.txt
-rwxrwxr-x. 1 kotmin kotmin 68 Apr 30 20:46 replace_with_time.sh
[kotmin@localhost DoPrezentacji]$ cat Kto_ma_co.txt
Ala ma psa od 2022-04-30
[kotmin@localhost DoPrezentacji]$
```

## Wskazówki:

- Do pobrania aktualnej daty możecie użyć programu date
  - Operator >> powoduje dopisanie do pliku zaś > spowoduje zapis danych
  - Nie wszystko od razu musi działać, nie bójcie się sięgnąć po wyszukiwarkę
- Psst. Upewnijcie się, że ignorowane są tylko rozszerzenia!**



# Podsumujmy



Czyli jak wprowadzając niejasności, tłumaczymy inne niejasności

# Nasz największy przyjaciel

## -git status

---

```
[kotmin@localhost Kotmin.github.io]$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore.swp

no changes added to commit (use "git add" and/or "git commit -a")
[kotmin@localhost Kotmin.github.io]$
```



# Czyli jednak można szybciej

```
$ git commit --am"zawieram w sobie git add ."
```

```
$ git config --global alias.ac „commit --am”  
$ git ac „jestem tym samym co wyżej”
```

Psst. Te nazwy zostały użyte, aby przykład był czytelniejszy podczas prawdziwego commitowania nie bawcie się z innymi w kalambury

\*kalambury są super! Ale po pracy, nie w kodzie

# Mv, rm czy git rm i git mv?

---



305



If you just use `rm`, you will need to follow it up with `git add <fileRemoved>`. `git rm` does this in one step.

You can also use `git rm --cached` which will remove the file from the index (staging it for deletion on the next commit), but keep your copy in the local file system.

[Share](#) [Improve this answer](#) [Follow](#)

answered Sep 15, 2011 at 16:51



**Andy**

41.5k ● 13 ● 67 ● 68



# Spojrzymy w przeszłość

## Git log

\$ git log

```
[kotmin@localhost Kotmin.github.io]$ ls
CSS          galeria.html  nbproject  src          zamowienie.html
formularze.html index.html    README.md  tabele.html
[kotmin@localhost Kotmin.github.io]$ git log
commit 6bd3f7add83a84ec1ae1d90f848ed2d0e5612185 (HEAD -> main)
Author: kotmin <kotmin@localhost.localdomain>
Date: Sat Apr 30 21:15:39 2022 +0200

    Added gitignore file

commit 344fb2822a401db6ffe2a49ced0f89c5d135a2a5 (origin/main, origin/HEAD)
Author: Kotmin <70173732+Kotmin@users.noreply.github.com>
Date: Sun Apr 24 07:23:34 2022 +0200

    Swap table

    Removing unnecessary comments. Swap tbody and tfoot.

commit 81f750310cf5c333caab1705098bc60437ab8277
Author: Kotmin <70173732+Kotmin@users.noreply.github.com>
Date: Sat Apr 23 06:34:13 2022 +0200

    137 line lucida grande

    Solved with:
    https://stackoverflow.com/questions/4375353/using-lucida-grande-in-windows

...skipping...
commit 6bd3f7add83a84ec1ae1d90f848ed2d0e5612185 (HEAD -> main)
Author: kotmin <kotmin@localhost.localdomain>
Date: Sat Apr 30 21:15:39 2022 +0200

    Added gitignore file

commit 344fb2822a401db6ffe2a49ced0f89c5d135a2a5 (origin/main, origin/HEAD)
Author: Kotmin <70173732+Kotmin@users.noreply.github.com>
Date: Sun Apr 24 07:23:34 2022 +0200

    Swap table
```

\$ git log --graph --oneline --decorate

```
[kotmin@localhost Kotmin.github.io]$ git log --graph --oneline --decorate
* 6bd3f7a (HEAD -> main) Added gitignore file
* 344fb28 (origin/main, origin/HEAD) Swap table
* 81f7503 137 line lucida grande
* 43906a3 242 line validator update
* 5784858 Changeing class of article
* 49527fe Small refactor added transparent class
* dc1ff6e Small refactor
* dbdec78 Setting margin_top
* 04b96da Adding css sheet
* d70c5a9 Adding CSS folder
* 6ebf7db Copy of 5_6_repo_without_css
* e24ac20 Initial commit
[kotmin@localhost Kotmin.github.io]$
```



# Proste cofanie czasu, dla amatorów

---

Zmiana nazwy ostatniego commita

Git commit –amend – „nowa nazwa”

Gdybyśmy zapomnieli dodać jakiś plik do commita

```
$ git add.
```

```
$ git commit –amend –no-edit
```

Tylko, że to działa tylko na lokalne repozytorium.

No chyba że zrobimy tak:

```
$ git push origin master --force
```

A pamiętacie jak mówiłem, że repozytorium prawie nie da się zepsuć? Za to słowo prawie w dużej mierze odpowiada flaga --force. Także używajcie jej proszę z rozwagą



# Czas na prawdziwą magię

---

Skok do konkretnego stanu projektu w czasie

```
$ git checkout identyfikator_commita
```

W najnowszym pull request wycofaj zmiany wprowadzone przez commit x

```
$ git revert id_commita
```

Faktyczne cofanie stanu repozytorium z wariacjami

```
$ git reset id_commita
```

Mamy tutaj jeszcze alternatywne tryby:

```
$ git reset --soft id_commita
```

```
$ git reset --hard id_commita
```



# A może by tak jeszcze jedna sztuczka?

---

```
$ git stash
```

Zapisuje lokalne zmiany w repozytorium, bez nadpisywania.

```
$ git stash pop
```

Przywraca nam te zmiany do kodu

Jeśli chcielibyśmy podnadużywać tego dobrodziejstwa to możemy używać wersji z dodawaniem nazw

```
$ git stash save nie_gotowe_liczenie_pensji
```

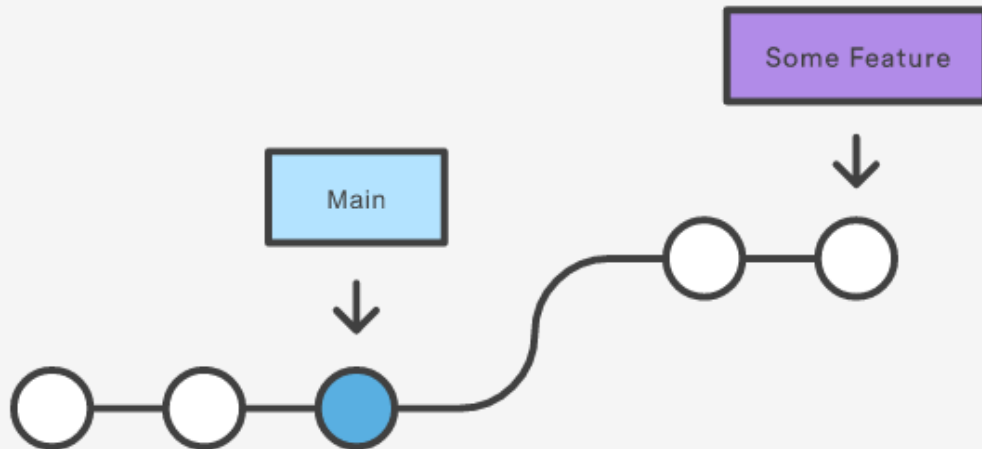
```
$ git stash list
```

```
$ git stash apply nie_gotowe_liczenie_pensji
```

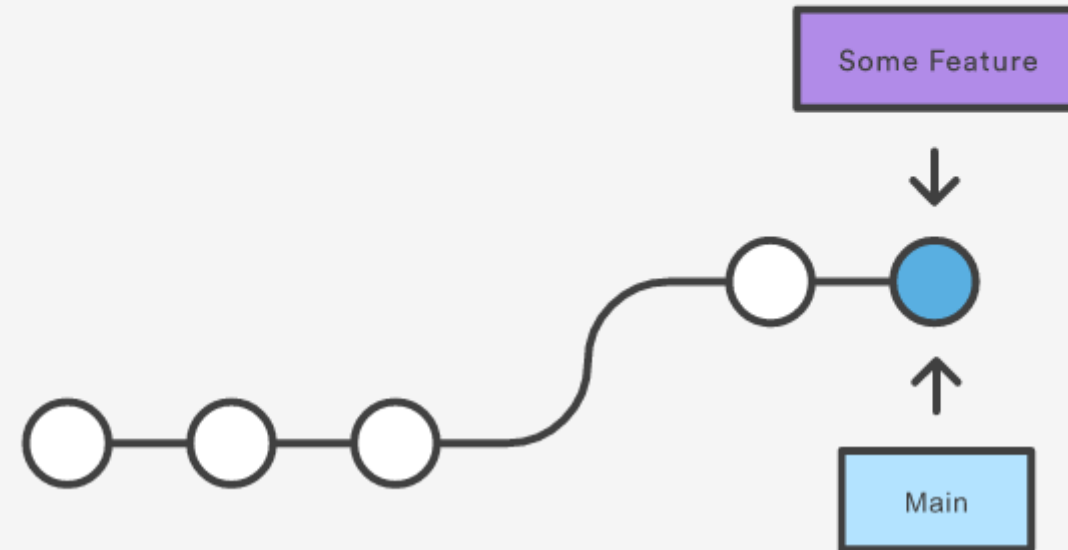


# Rodzaje łączeń merge

Before Merging



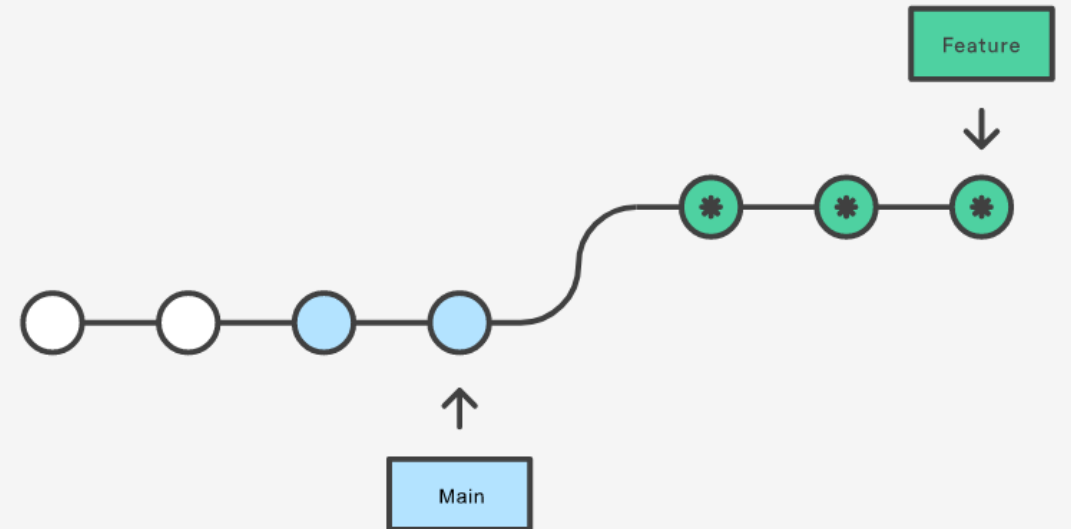
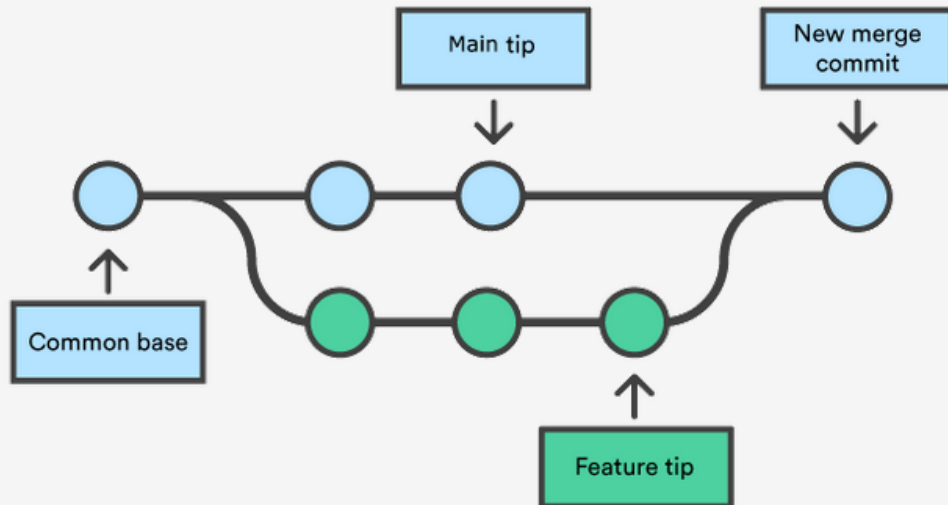
After a Fast-Forward Merge



# Squash vs rebase

[Main] \$ git merge --squash feature  
[Main] \$ git rebase feature

Zapis następuje do branch'a, z którego wywołujemy komendę, w naszym przypadku będzie to Main.





# Git checkout

---

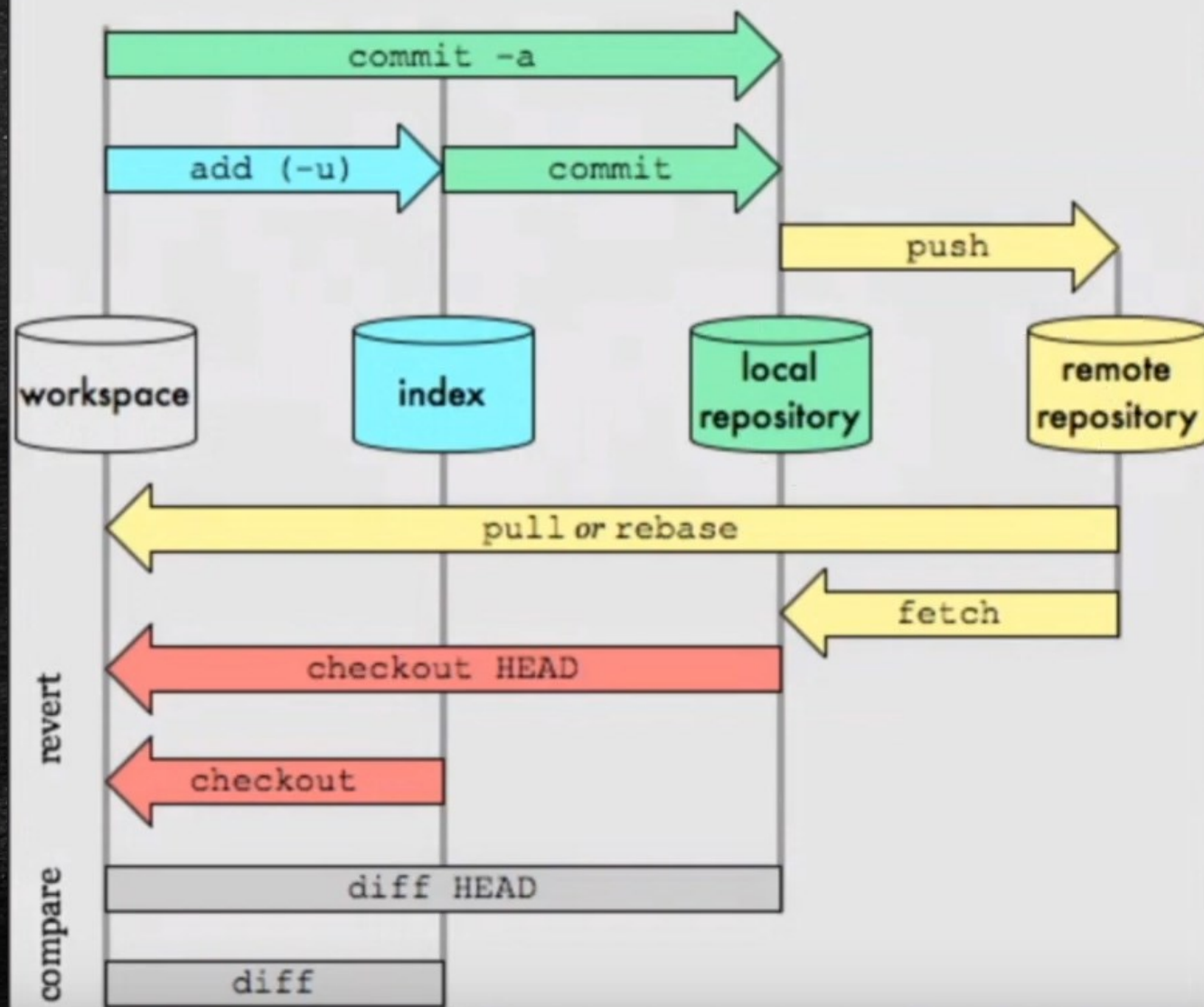
Skok do poprzednio używanej gałęzi:  
\$ git checkout -

Stworzenie nowej gałęzi:  
\$ git checkout -b feature



# Git Data Transport Commands

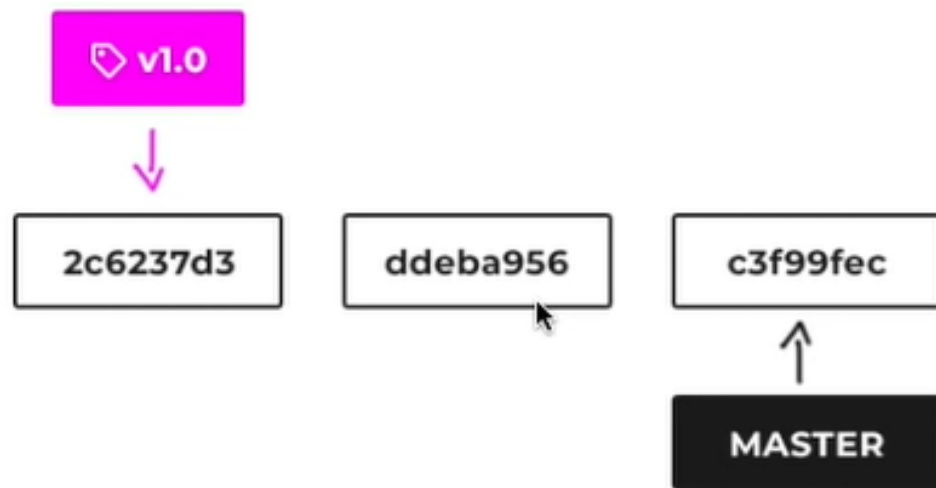
<http://osteele.com>





# Czym są tagi?

Możliwość etykietowania konkretnych commit'ów.



## Semantic Versioning 2.0.0

### Summary

Given a version number MAJOR.MINOR.PATCH, increment the:

1. MAJOR version when you make incompatible API changes,
2. MINOR version when you add functionality in a backwards compatible manner, and
3. PATCH version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

<https://semver.org/>

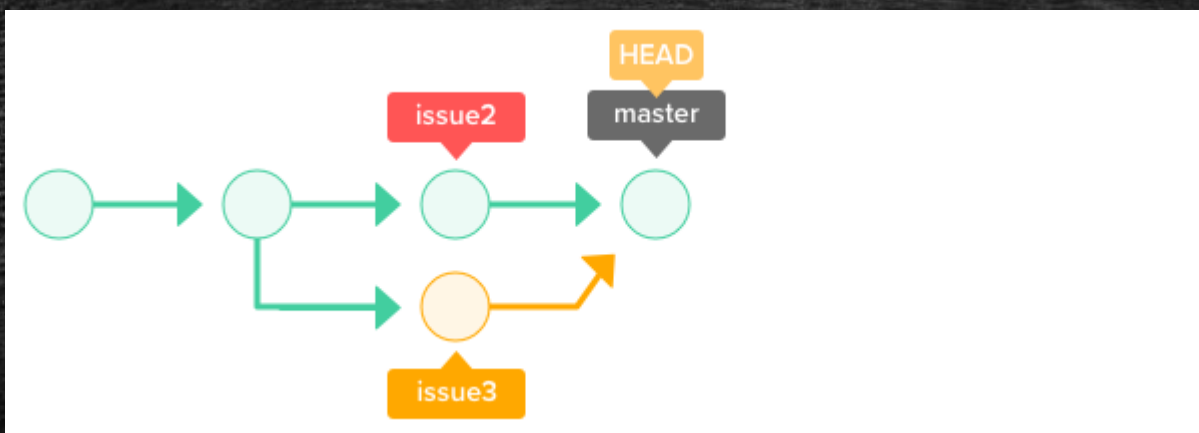
# Najważniejsze polecenia `git tag`

---

- Wyświetlanie wszystkich tagów `$ git tag`
- Podgląd tagu `$ git tag numer_tagu`
- Usuwanie tagu `$ git tag -d numer_tagu`
- Utworzenie tagu `$ git tag v5.0 -a -m „Tag mess”`
- Utworzenie tagu do dowolnego commitu `$ git tag v5.0 hash-commitu -a -m „Tag message”`
- Wysyłanie tagów do zdalnego repozytorium `$ git push --tags`
- Usuwanie tagu ze zdalnego repo `$ git push nazwa_repo -d v5.0`



# Merge conflict



```
1 require('./bootstrap');
2
3 window.Vue = require('vue');
4
5 Vue.component('example-component', require('./components/
  • ExampleComponent.vue').default);
6
7 <==== HEAD
8 Vue.component('follow-button', require('./components/
  • FollowButton.vue').default);
9 Vue.component('profile-attributes', require('./components/
  • ProfileAttributes.vue').default);
10 Vue.component('profile-posts', require('./components/
  • ProfilePosts.vue').default);
11
12 =====
13 Vue.component('image-editor', require('./components/
  • ImageEditor.vue').default);
14 Vue.component('image-uploader-field', require('./components/
  • ImageUploaderField.vue').default);
15 Vue.component('create-post', require('./components/
  • CreatePost.vue').default);
16 Vue.component('image-viewer', require('./components/
  • ImageViewer.vue').default);
17
18 >>>>>> 812a998bac9fc14f25542d30a95df9da7bdeccf8
19
20 Use me our changes
21
22
23
24
25
26
27
28 const app = new Vue({
29   el: '#app',
30 });
31
```

# Rozwiązywanie konfliktów

---

Najczęściej używane komendy do rozwiązywania konfliktów:

- `git merge --abort`
- `git log --merge`
- `git diff`
- `git reset --mixed`



# Ćwiczenie

## Rozwiązać merge conflict

```
→ git:(master) git merge another_feature  
CONFLICT (modify/delete): index.html deleted in HEAD and modified in an  
other_feature. Version another_feature of index.html left in tree.  
Automatic merge failed; fix conflicts and then commit the result.
```

- 1) Utworzenie dwóch branchy zawierających ten samy plik o różnej zawartości
- 2) Merge dwóch branchy
- 3) Rozwiązywanie konfliktu
- 4) Zapisywanie zmian



















## Dobre zasady dotyczące współpracy

---

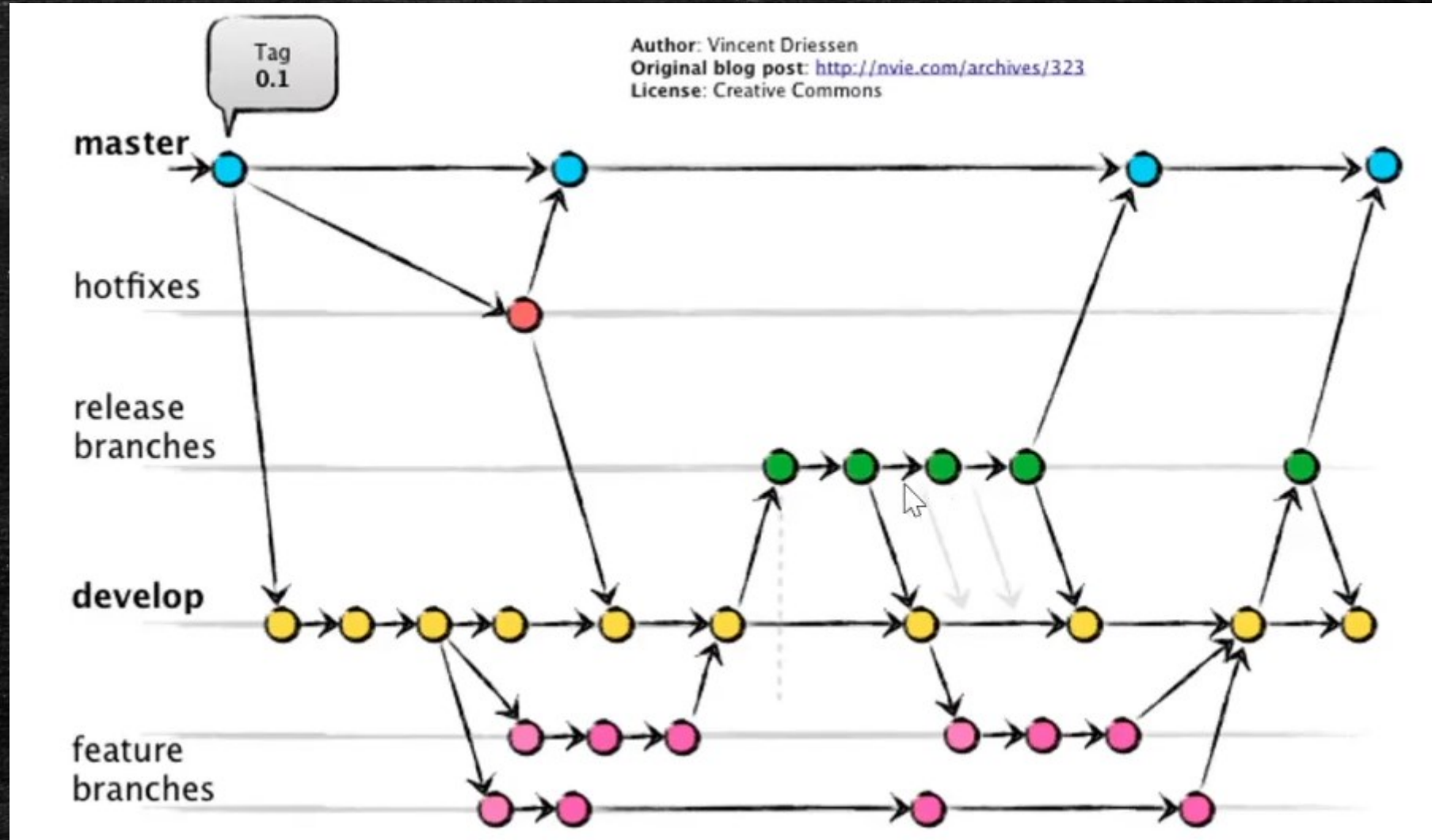
- Ściśle trzymaj się ustaleń dotyczących struktury i nazewnictwa w projekcie
- Dbaj o czytelność swojego kodu
- Korzystając z zewnętrznych bibliotek upewnij się, że wiesz jak nimi zarządzać
- Licencja oprogramowania powinna być dopasowana do potrzeb projektu
- Twórz dokumentację



# Struktura projektu

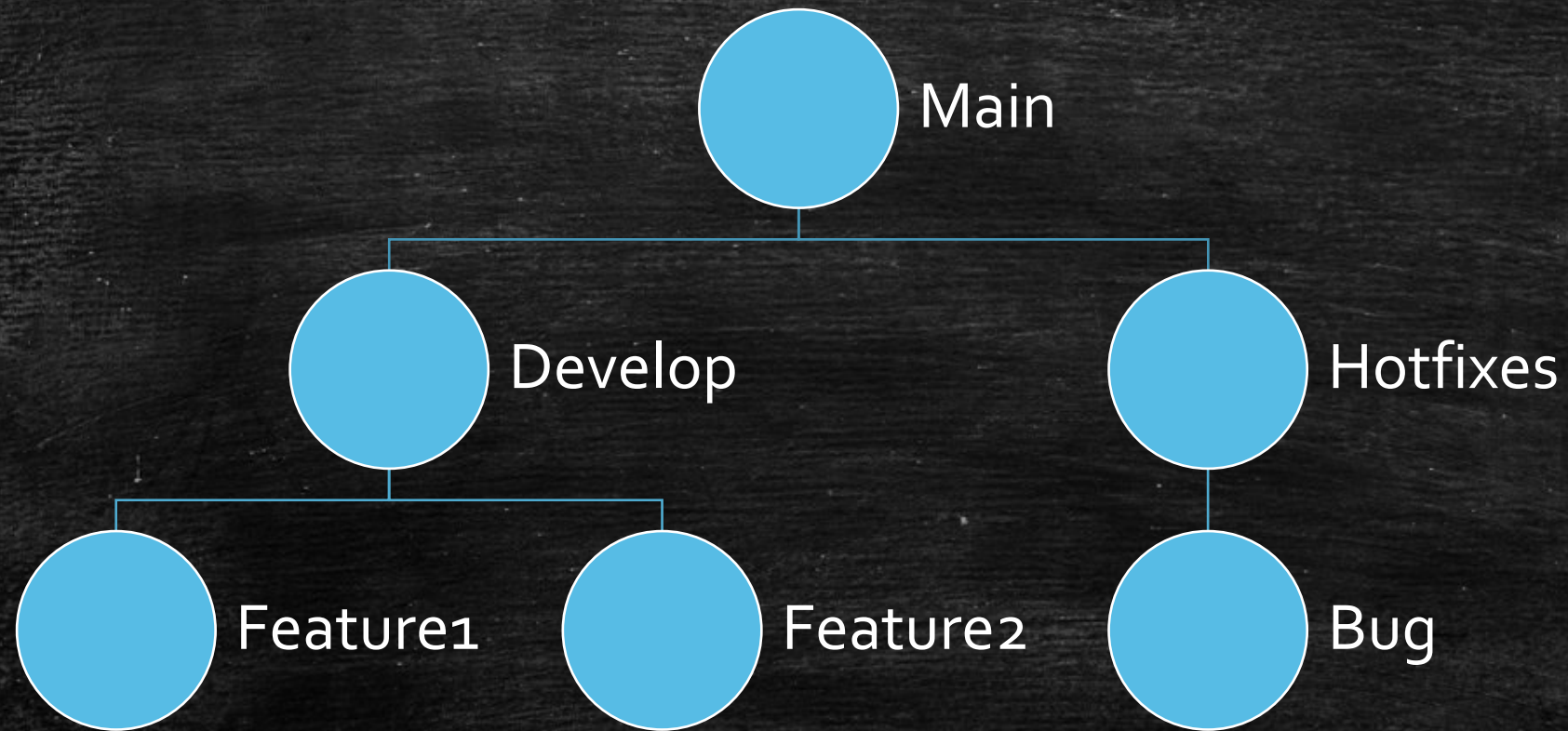
 dvgcode Initial commit	59b341f on 6 Apr	 1 commit
 .config	Initial commit	last month
 .github	Initial commit	last month
 build	Initial commit	last month
 dep	Initial commit	last month
 doc	Initial commit	last month
 res	Initial commit	last month
 samples	Initial commit	last month
 src	Initial commit	last month
 test	Initial commit	last month
 tools	Initial commit	last month
 .dockerignore	Initial commit	last month
 .editorconfig	Initial commit	last month
 .gitattributes	Initial commit	last month
 .gitignore	Initial commit	last month

# Git flow

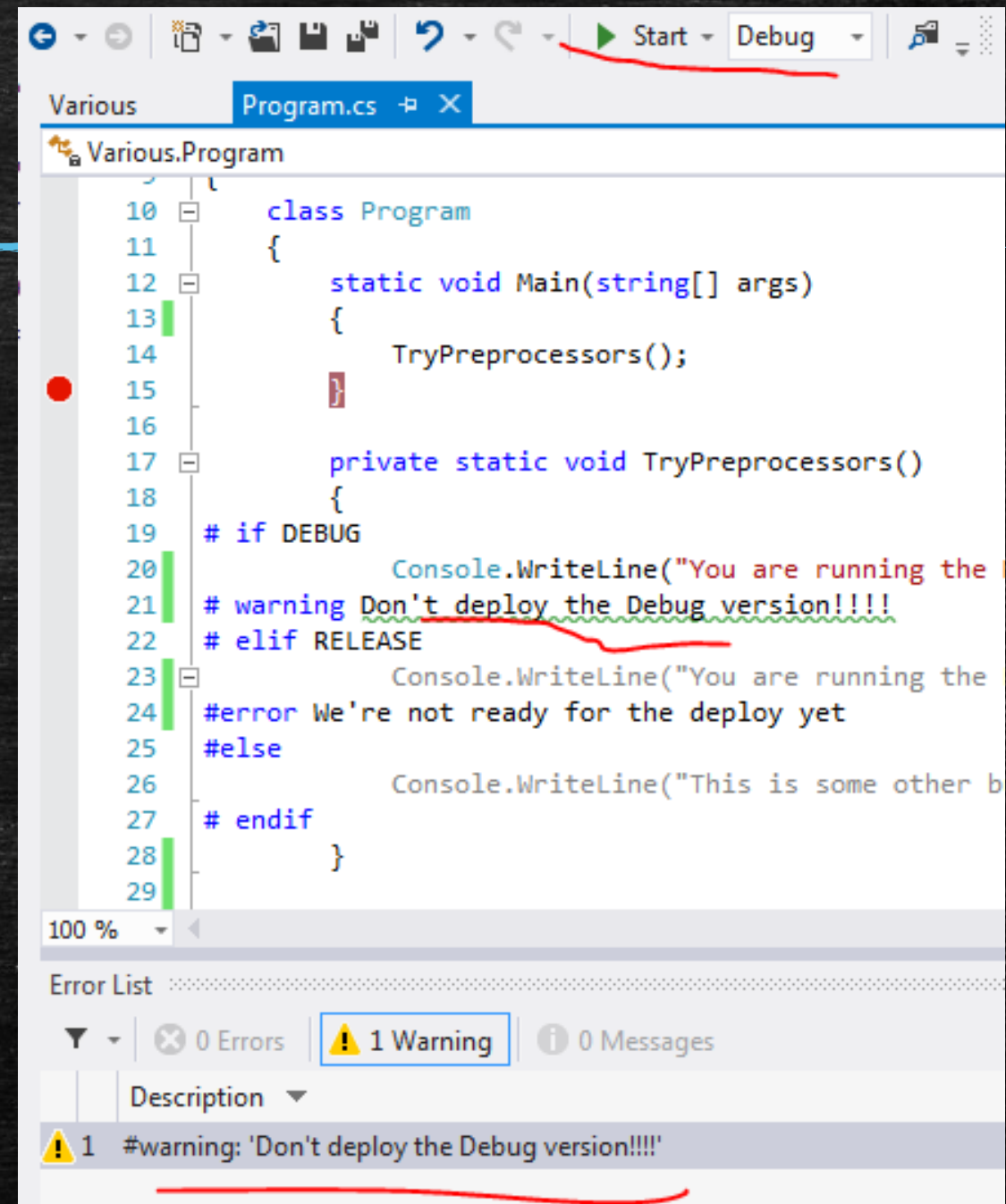




# Git Flow



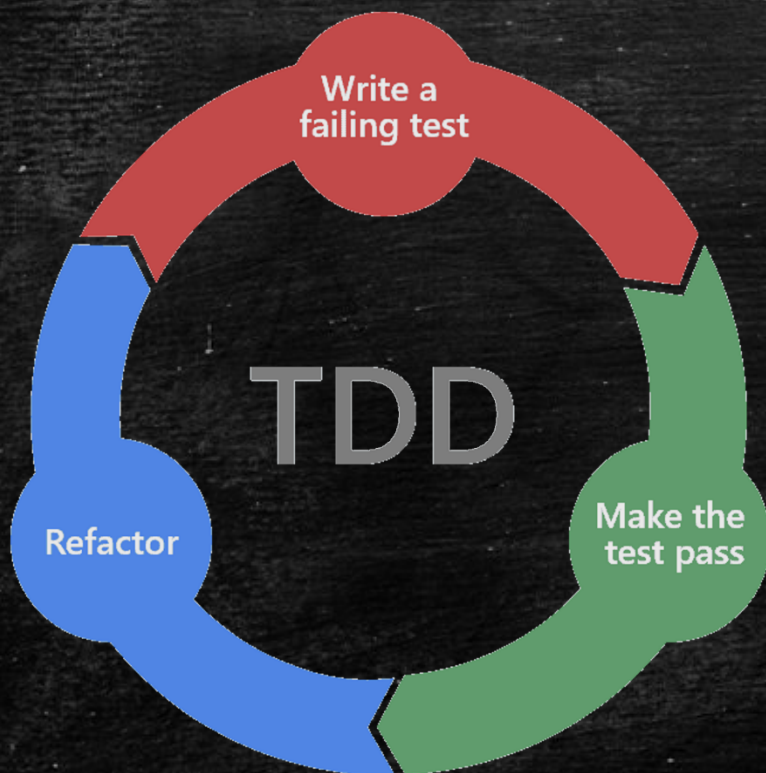
Kompiluje się == działa?





# Testowanie, a po co to komu?

---

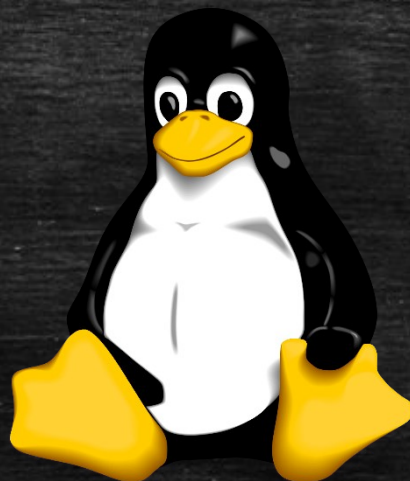


Testy zapewniają:

- Lepszą jakość kodu
- Bezpieczeństwo wprowadzania zmian
- Płynność tworzenia kodu

# GitHub Actions

---



To swoistego rodzaju system reagowania na wydarzenia. Na przykład

if ktoś próbuje git push:  
zareaguj\_wg\_reguł()

```
on:  
  push:  
    branches: [ main ]  
  pull_request:  
    branches: [ main ]
```



main ▾ [Ala\\_i\\_jej\\_zwierzeta](#) / [.github](#) / workflows /

Go to file

Add file ▾

...



NotFunnyStudent Create python-package.yml

● in 2 days ⌚ History

..



python-package.yml

Create python-package.yml



NotFunnyStudent Create Kto\_ma...



in 2 days



5



.github/wo...

Create python-package.yml

now



test

Adding test file

now



.gitignore

Initial commit

now



Kto\_ma\_c...

Create Kto\_ma\_co.txt

now



README....

Initial commit

now



main.py

Add main

now



Create Kto\_ma\_co.txt

Python package #4: Commit 5057fdb pushed to  
NotFunnyStudent



Adding test file

Python package #3: Commit 8b3a643 pushed to  
NotFunnyStudent

main ▾

Ala\_i\_jej\_zwierzeta / test / test\_main.py / <> Jump to ▾

Go to file

...



NotFunnyStudent Adding test file ✖

Latest commit 8b3a643 in 2 days  History

1 contributor

6 lines (5 sloc) | 173 Bytes

Raw

Blame



```
1  import pytest
2
3  def test_tstring_comparsion():
4      with open('Kto_ma_co.txt') as f:
5          to_cmp=f.read()
6      assert to_cmp.strip()=="Ala ma psa od 2022-04-30"
```

```
1  Ala ma psa od 2022-04-30
```



# Jeszcze raz po co?

```
12 test/test_main.py F [100%]
13
14 ===== FAILURES =====
15 _____ test_tstring_comparsion _____
16
17     def test_tstring_comparsion():
18         with open('Kto_ma_co.txt') as f:
19             to_cmp=f.read()
20 >     assert to_cmp.strip()=="Ala ma psa od 2022-04-30"
21 E     AssertionError: assert 'Ala ma koalę od 2022-04-30' == 'Ala ma psa od 2022-04-30'
22 E         - Ala ma psa od 2022-04-30
23 E           ?      ^^
24 E         + Ala ma koalę od 2022-04-30
25 E           ?      ^^ ++
26
27 test/test_main.py:6: AssertionError
28 ===== short test summary info =====
29 FAILED test/test_main.py::test_tstring_comparsion - AssertionError: assert 'A...
30 ===== 1 failed in 0.03s =====
31 Error: Process completed with exit code 1.
```

# Ćwiczenie

Korzystając z dostępnych wam źródeł dołączcie bibliotekę testującą. Możecie użyć dowolnej biblioteki i dowolnego języka. Skonfigurujcie również repozytorium na GitHubie tak aby Actions przeprowadzało sprawdzenie przy zdarzeniach on\_push oraz merge

```
12  test/test_main.py .  
    [100%]  
13  
14  ===== 1 passed in 0.01s  
    =====
```

Wskazówki:

- Można skorzystać z predefiniowanych opcji jakie oferuje github
- Pliki konfiguracyjne zakładki Actions powinny znaleźć się w  
,,/.github/workflows/

Psst. Możecie też oprzeć się na naszym szablonie git clone ...



# Linkografia do GitHub Actions

---

Jak napisać pierwszy test?

<https://youtu.be/upoLSgFhuEk>

Wstęp do korzystania z pytest oraz GitHub Actions:

<https://youtu.be/DhUpkWjOhME>

Oficjalna dokumentacja pytest:

<https://docs.pytest.org/en/7.1.x/>



pytest

# Co jeszcze oferuje nam GitHub?

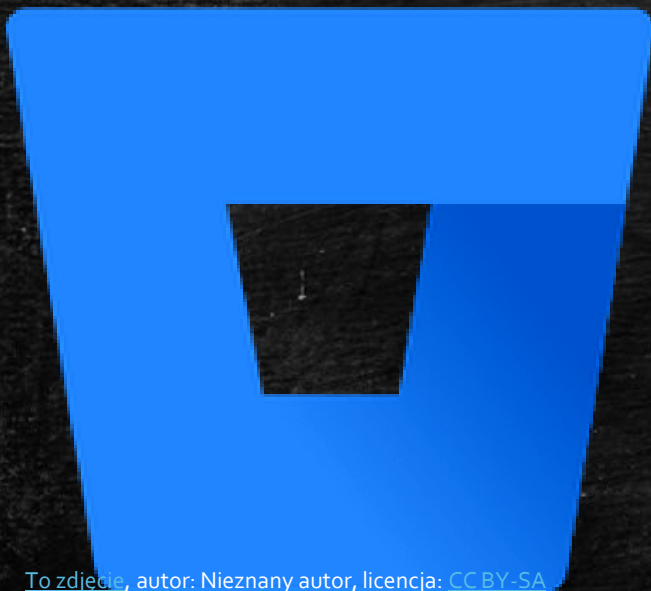
---





# Oprogramowanie do pracy z gitem

---



To zdjęcie, autor: Nieznany autor, licencja: CC BY-SA



GitLab



GitKraken

Atlassian



SourceTree



# Na co jeszcze warto spojrzeć?

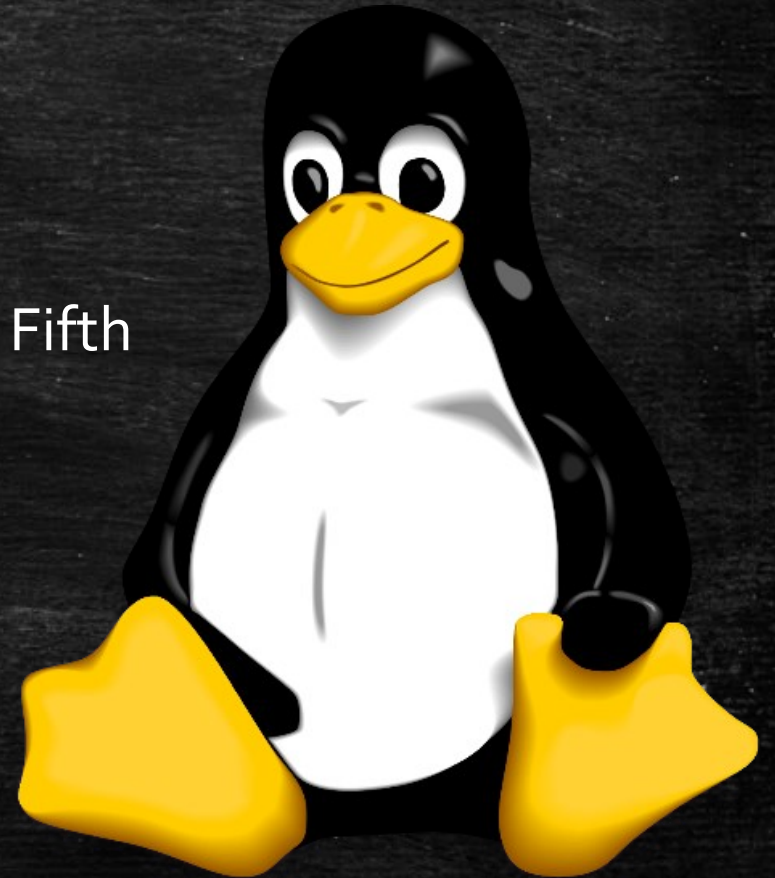
---

## Literatura:

- Clean Code- Robert C. Martin
- Clean Architecture-Robert C.Martin
- UNIX and Linux System Administration Handbook Fifth Edition – E.Nemeth,G.Snyder ...
- Bezpieczeństwo aplikacji webowych- Securitum
- The Mythical Man-Month: Essays on Software Engineering- Brooks P. Frederick
- Pro Git, 2nd edition-Chacon Scott, Straub Ben

## Linki:

- <https://www.atlassian.com/pl/git/tutorials>
- <https://training.github.com/>
- <https://git-scm.com/docs>





# HIRE ME!

- O rozmowach kwalifikacyjnych słów kilka



# Szczególne podziękowania kierujemy do

---

- dr Michał Dolecki
- dr Maciej Pańczyk
- Aleksander Chotecki
- Oraz wielu ludzi dobrej woli, którzy tworzyli tutoriale w internecie



Czy macie państwo jakieś pytania?



# Dziękuję za uwagę



Prezentowali:  
Paweł Jan Tłusty  
Viktor Vodnev

Życzymy smacznej kawusi