

Kreatywne programowanie

Zajęcia nr 1

GIT - System Kontroli wersji
<https://git-scm.com>



Piotr Radzikowski

ptr.radzikowski@gmail.com

radzikowski.one.pl

Organizacja pracy przy projekcie

- trello.com
- facebook.com
- git (github.com)

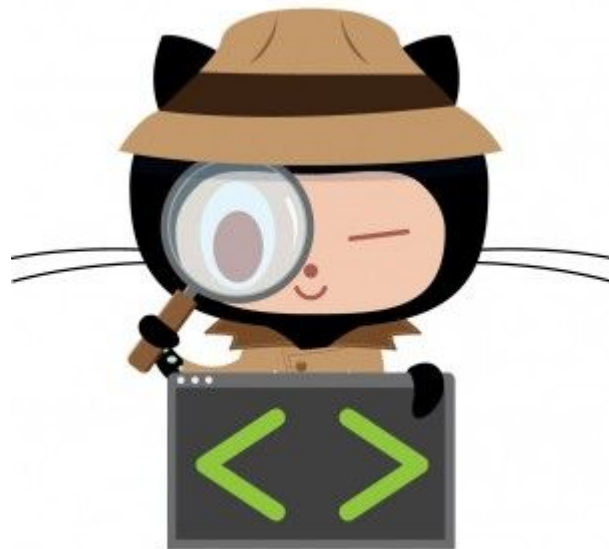
Git jest git

Ściągnijmy gita ze strony głównej projektu <https://git-scm.com>



VSC - Version Control System

***System Kontroli Wersji** śledzi wszystkie zmiany dokonywane na pliku (lub plikach) i umożliwia przywołanie dowolnej wcześniejszej wersji.*



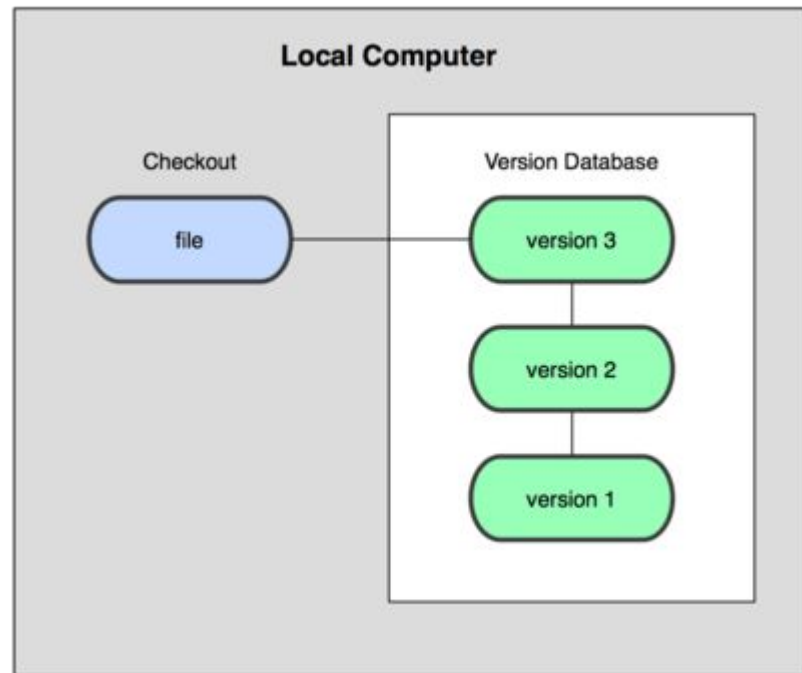
Wyróżniamy kilka typów systemów kontroli wersji

- lokalny system kontroli wersji
- scentralizowany system kontroli wersji
- rozproszony system kontroli wersji



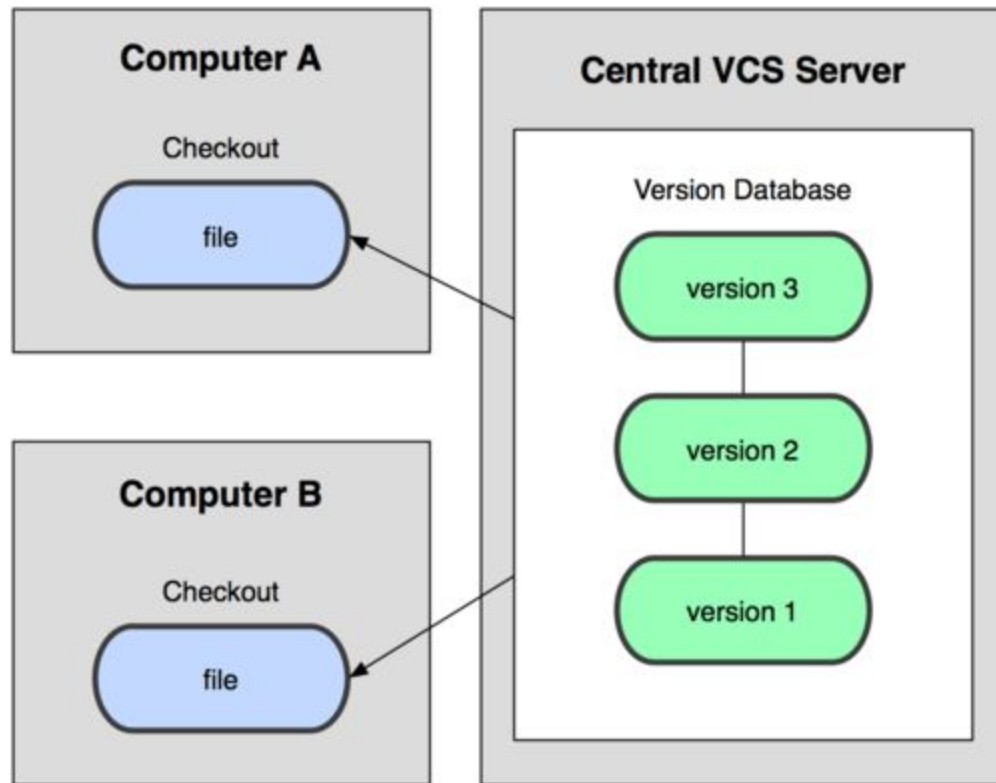
Local Control System Version

Metoda kontroli wersji polegająca na kopiowaniu plików do innego katalogu (może nawet oznaczonego datą, jeśli są sprytni).



CVCS - Central Version Control System

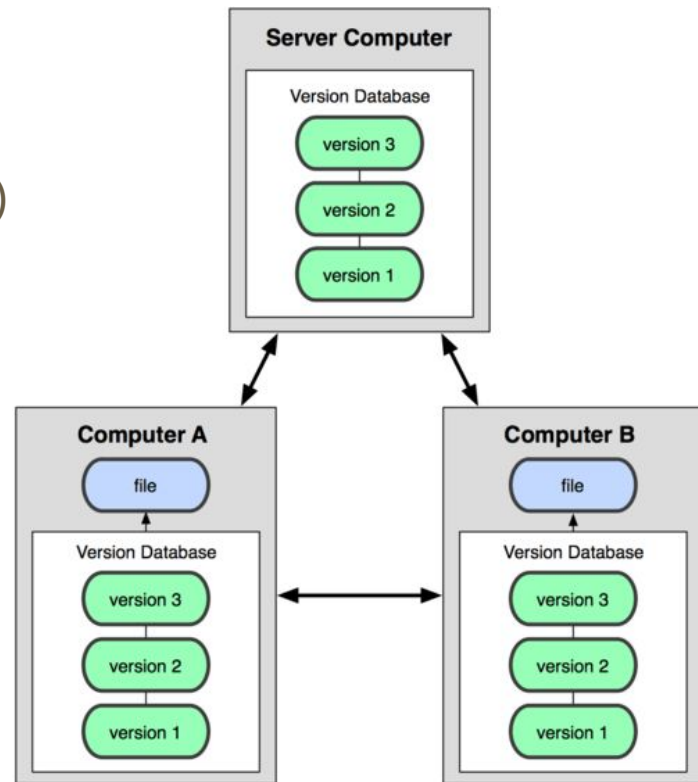
Przykładem jest Subversion(SVN)
lub TFS czy kiedyś bardzo
popularny CVS



DVCS - Distributed Version Control System

Rozproszony system kontroli wersji
(DVCS - Distributed Version Control System).
Przykładem jest np. Git i Mercurial (zwany też HG)

Każda maszyna przechowuje całe repozytorium



GIT

- rozproszony system kontroli wersji (DVCS)
- “narodził się” 7 kwietnia 2005 roku
- autor: Linus Torvalds
- rewolucja w systemie kontroli wersji



Instalacja gita

- zaznaczenie opcji integracji z domyślny cmd

Konfiguracja gita

Zmienne identyfikacyjne i konfiguracja gita

- przełącznik "global"

```
x git config --global user.email "your email@example.com"█
```

```
x git config --global user.name "Imie Nazwisko"█
```

Repozytorium (potocznie: repo)

Katalog zawierający wszystkie informacje o aktualnym stanie projektu i jego historii.

Może zawierać odwołania do innych repo.



shutterstock.com • 1024719589

git init

Tworzenie (inicjalizacja) projektu opartego na git, możliwe jest wykonanie tej operacji w dowolnym katalogu, pustym lub z istniejącym już projektem.

```
radzikowski@radzikowski-win MINGW64 ~/development/01-git
$ git init
Initialized empty Git repository in C:/Users/radzikowski/development/01-git/.git
/
radzikowski@radzikowski-win MINGW64 ~/development/01-git (master)
```

README.md

markdown - pseudo język do formatowania tekstu.

```
It's very easy to make some words bold and other words  
*italic* with Markdown. You can even  
[link to Google!](http://google.com)
```

It's very easy to make some words **bold** and other words *italic* with
Markdown. You can even [link to Google!](http://google.com)

<https://guides.github.com/features/mastering-markdown/>

README.md

Plik opisuje jak korzystać z projektu, oraz dodatkowe przydatne informacje na temat projektu, np.

- jak go uruchomić na komputerze
- jak uruchomić testy
- z czego składa się projekt (biblioteki)

README.md Raw

Project Name

TODO: Write a project description

Installation

TODO: Describe the installation process

Usage

TODO: Write usage instructions

Contributing

1. Fork it!
2. Create your feature branch: `git checkout -b my-new-feature`
3. Commit your changes: `git commit -am 'Add some feature'`
4. Push to the branch: `git push origin my-new-feature`
5. Submit a pull request :D

History

TODO: Write history

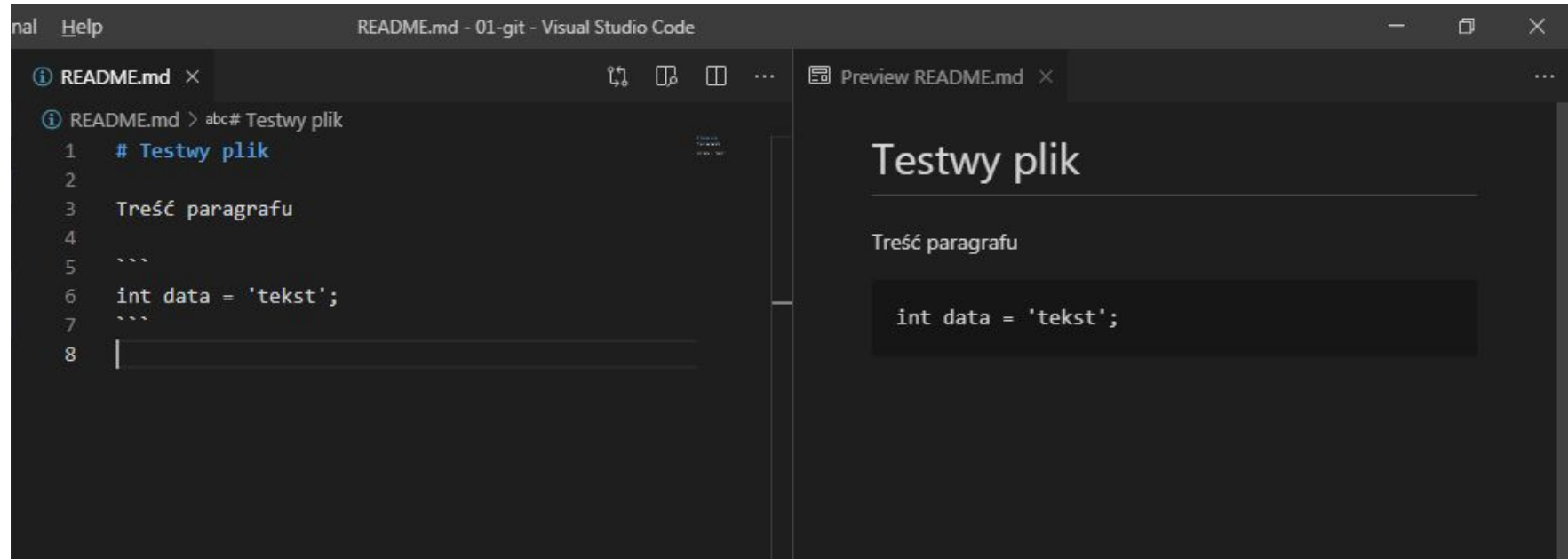
Credits

TODO: Write credits

License

TODO: Write license

<https://guides.github.com/features/mastering-markdown/>



git status

Polecenie do ukazania zmian plików wraz z kolorowaniem ukazującym jakie pliki zostały zmienione.

```
→ 01-git git:(master) ✕ git status
On branch master

No commits yet

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

    readme.md

nothing added to commit but untracked files present (use "git add" to track)
→ 01-git git:(master) ✕
```

git add

Polecenie służy do dodawania pliku/plików do naszego repozytorium.

```
→ 01-git git:(master) ✕ git add
Nothing specified, nothing added.
Maybe you wanted to say 'git add .'?
```

```
→ 01-git git:(master) ✕ git add readme.md
→ 01-git git:(master) ✕
```

```
→ 01-git git:(master) ✕ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        nowy plik:       readme.md

→ 01-git git:(master) ✕
```

git commit

Zapisanie wersji względem zmian, tworzenie "historii" zmian w repozytorium

```
→ 01-git git:(master) x git commit
```

```
→ 01-git git:(master) git commit -m "init readme file"
```

```
x git commit . -m "init readme file"
```

```
x git commit .
```

core.editor

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

Basic Vim Commands

- `:w` Write the current file
- `:wq` Write the current file and exit.
- `:q!` Quit without writing
- To change into insert mode: `i` or `a`
 - Use escape to exit
- search forward `/`, repeat the search backwards: `N`
- Basic movement:
 - `h l k j` character left, right; line up, down (also arrow keys)
 - `b w` word/token left, right
 - `ge e` end of word/token left, right
 - `0 $` jump to first/last character on the line
- `x` delete
- `u` undo

<https://wiki.gentoo.org/wiki/Vim/Guide> and <http://tnerual.eriogerg.free.fr/vimqrc.pdf>



git log

Historia kto/co/kiedy?

można prześledzić proces powstawania każdej funkcji

```
commit 2842215e343b7418851fd03785c978841057544d (HEAD -> master)
```

```
Author: Piotr Radzikowski <ptr.radzikowski@gmail.com>
```

```
Date:   Fri Oct 4 14:39:35 2019 +0200
```

```
    add push and pull description
```

```
commit d09df0625294d769db6b2bfc56c61ecf0836d566
```

```
Author: Piotr Radzikowski <ptr.radzikowski@gmail.com>
```

```
Date:   Fri Oct 4 14:37:35 2019 +0200
```

```
    Fix readme file - add basic git commands
```

```
commit c8e105f24db8f58f4723ba31478c4f59c3e0334a
```

```
Author: Piotr Radzikowski <ptr.radzikowski@gmail.com>
```

```
Date:   Fri Oct 4 14:34:35 2019 +0200
```

```
    fix Readme title typo
```

```
commit 009eb546f28130ce0fd61c6646b2f570726290a6
```

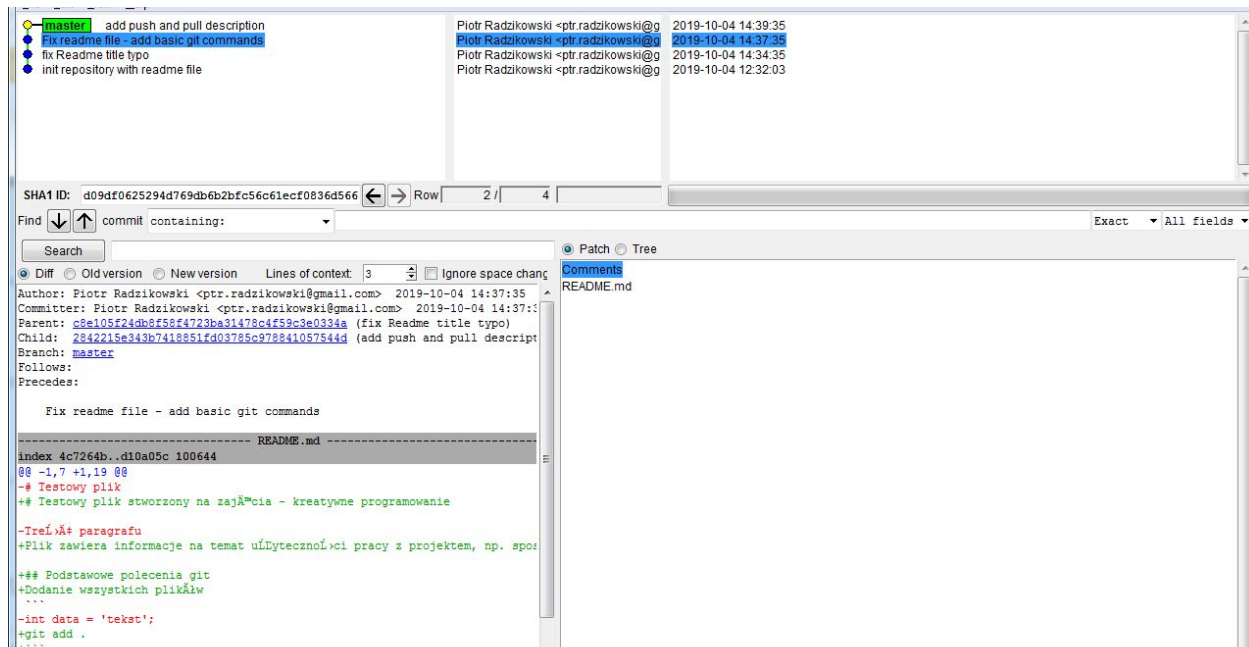
```
Author: Piotr Radzikowski <ptr.radzikowski@gmail.com>
```

```
Date:   Fri Oct 4 12:32:03 2019 +0200
```

```
    init repository with readme file
```



graficzny interfejs do reprezentacji historii i gałęzi* gita



git diff

```
x git diff
```

```
diff --git a/readme.md b/readme.md
index 3d79c13..66d5f7a 100644
--- a/readme.md
+++ b/readme.md
@@ -1,6 +1,13 @@
-# test
-asd
+# Testowy projekt GIT
+Projekt powstał na potrzeby zajęć z kreatywnego programowania

+## podstawowe komendy
+```
+var code = 'abc';
+git commit -m "wiadomosc"
+```
+git push
+```
+git pull
+```
\ No newline at end of file
(END)
```

git push

Komenda do wysyłania zmian które zostały przez nas zapisane (za-commit-owane) w naszym projekcie, do katalogu w internecie*

```
C:\Users\radzikowski\development\01-git>git push
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using

    git remote add <name> <url>

and then push using the remote name

    git push <name>
```


Zanim zapiszemy efekty naszych prac

Musimy wskazać dla gita (git-owi :)) gdzie te zmiany muszą być wysłane



Portale które wspierają protokół GIT

- github.com
- bitbucket.org
- gitlab.com

The screenshot displays the GitLab pricing page at about.gitlab.com/pricing/. The page features three main pricing tiers: Free, Bronze, and Silver. Each tier includes a description of its capabilities, a price per user per month, and a list of features. The Free tier is available at no cost, while the Bronze and Silver tiers are paid plans. The page also includes a navigation bar with links to Product, Pricing, Resources, Blog, Support, and Jobs. A 'Self-hosted' option is visible on the right side of the page.

Free	Bronze	Silver
Helping developers build, deploy, and run their applications.	Enabling teams to speed DevOps delivery with automation, prioritization, and workflow.	Enabling IT to scale DevOps delivery with progressive deployment, advanced configuration, and consistent standards.
\$0 per user per month	\$4 per user per month (billed annually)	\$19 per user per month (billed annually)
Sign Up	Buy Now	Buy Now
2,000 CI pipeline minutes per group per month on our shared runners	2,000 CI pipeline minutes per group per month on our shared runners	10,000 CI pipeline minutes per group per month on our shared runners
Unlimited private and public projects and unlimited collaborators	All features from Free and: <ul style="list-style-type: none">→ Next business day Support→ Multiple approvers in code review→ Merge approvals→ Code Quality	All features from Bronze and: <ul style="list-style-type: none">→ Multiple Group Issue Boards→ Priority Support→ Multi-project pipeline graphs→ Deploy Boards→ <u>Timed and manual incremental</u>

Quick setup — if you've done this kind of thing before

or **HTTPS** **SSH** `git@github.com:radzikowski/01-kreatywne-programowanie.git`



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# 01-kreatywne-programowanie" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:radzikowski/01-kreatywne-programowanie.git
git push -u origin master
```



...or push an existing repository from the command line

```
git remote add origin git@github.com:radzikowski/01-kreatywne-programowanie.git
git push -u origin master
```



...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

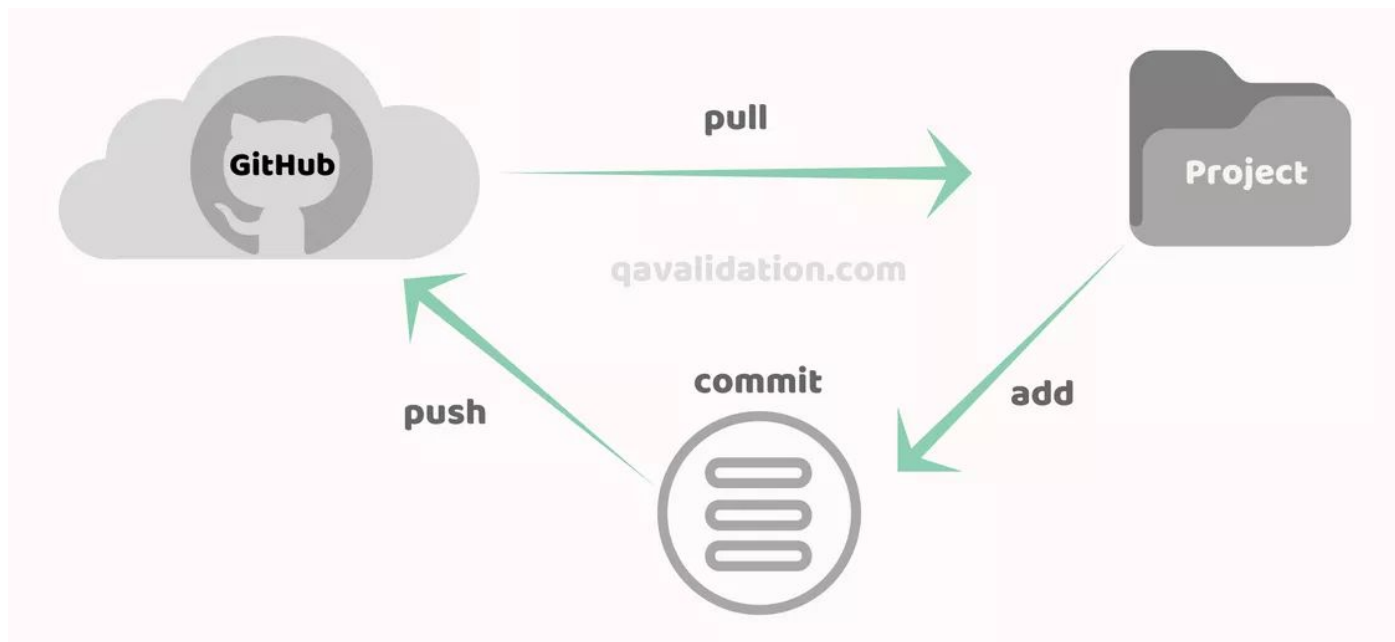
[Import code](#)

git push

Wysłanie zmian na nasze repozytorium.

git pull

Pobranie zmian z naszego repozytorium



Zapamiętanie hasła w systemie

- **Windows:** Git Windows Credentials Manager
 - `git config --global credential.helper wincred`

ssh-keygen

<https://help.github.com/en/enterprise/2.16/user/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

Generating a new SSH key

- 1 Open Git Bash.
- 2 Paste the text below, substituting in your GitHub Enterprise email address.

```
$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

This creates a new ssh key, using the provided email as a label.

```
> Generating public/private rsa key pair.
```

Podsumowanie

Poznaliśmy komendy:

- git init
- git status
- git add
- git commit
- git log
- gitk
- git diff
- git push
- git pull

In case of fire



1. git commit



2. git push



3. leave building

Dziękuję za uwagę!
Co na następnych zajęciach?



Piotr Radzikowski



ptr.radzikowski@gmail.com



radzikowski.one.pl