

# POLITECHNIKA LUBELSKA WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI

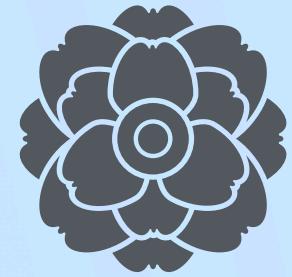
# Programowanie Aplikacji w Chmurze Obliczeniowej

# Laboratorium nr 6

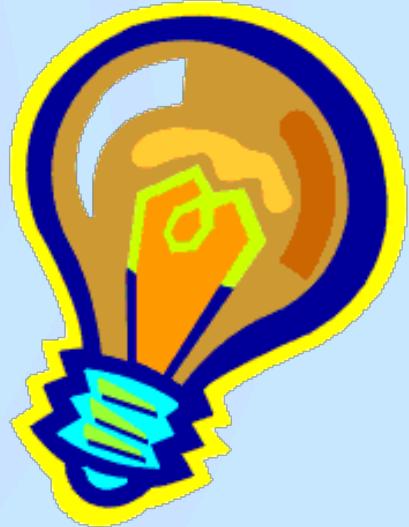
Konfiguracja i wykorzystanie klienta CLI dla usług Github. Rejestr obrazów ghcr.io - zasady konfiguracji i korzystania. Deklarowanie i użycie secrets w plikach Dockerfile oraz w procesie budowania obrazów. Zasady tworzenia rozszerzonych frontend-ów.

Dr inż. Sławomir Przyłucki  
[s.przylucki@pollub.pl](mailto:s.przylucki@pollub.pl)





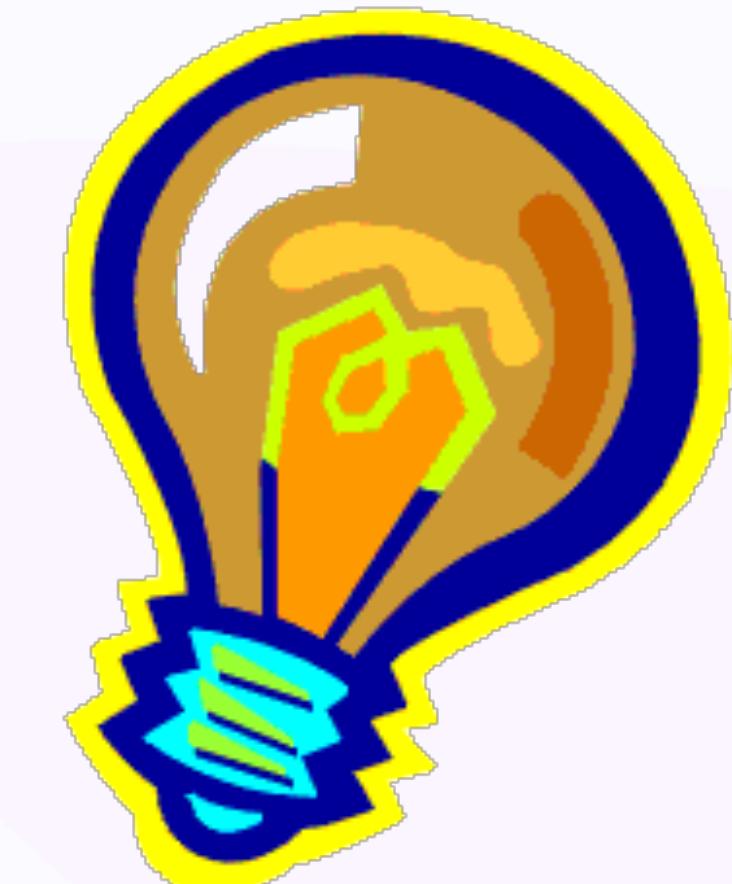
## Obsługa Github za pomocą klienta CLI - wprowadzenie - cz. I

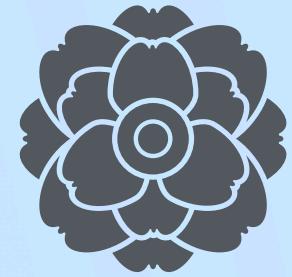


Przed rozpoczęciem działań opisanych w tej instrukcji **KONIECZNIE** należy zagwarantować, że w systemie jest zainstalowane (i skonfigurowane) narzędzie git

W wielu przyszłych ćwiczeniach laboratoryjnych będzie wykorzystywane repozytorium Github (np. podczas deklarowania łańcuchów CI/CD) i dlatego warto już teraz skonfigurować środowisko programistyczne do potrzeb realizacji przyszłych zadań.

- Studenci korzystający z komputerów będących na wyposażeniu sal laboratoryjnych powinni **KONIECZNIE**, po zakończeniu zajęć, usunąć wygenerowane klucze i tokeny dostępowe oraz wylogować się z wykorzystywanych serwisów.
- W/w klucze i tokeny **NALEŻY** zachować w bezpiecznych miejscach np. na dyskach sieciowych. Można też wykorzystać przenośne nośniki pamięci, np. Dyski, pen-y itp.





## Obsługa Github za pomocą klienta CLI - wprowadzenie - cz. II

Dostęp do serwisu Github może być realizowany na kilka sposobów. Szczegółowy opis poszczególnych metod znajduje się w dokumentacji Github:

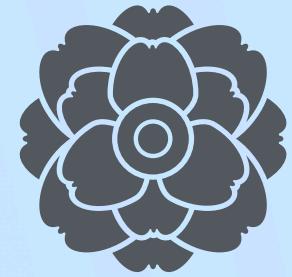
<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/about-authentication-to-github>



Ponieważ wykorzystywane będą głównie dwa metody autoryzacji, w oparciu o: klucze dla połączeń SSH oraz osobiste tokeny dostępowe PAT (ang. Personal Access Token) to szczególnie zalecam zapoznanie się z w/w dokumentacją właśnie w tym zakresie.



W trakcie laboratorium zainstalowane będzie narzędzie do zarządzania repozytoriami Github z poziomu interfejsu CLI (terminala tekstowego). Proszę pamiętać, że narzędzie to **NIE ZASTĘPUJE** polecen oferowanych przez program git a jedynie stanowi ich **UZUPEŁNIENIE** podczas korzystanie z usług Github,



## Obsługa Github za pomocą klienta CLI - wprowadzenie - cz. II



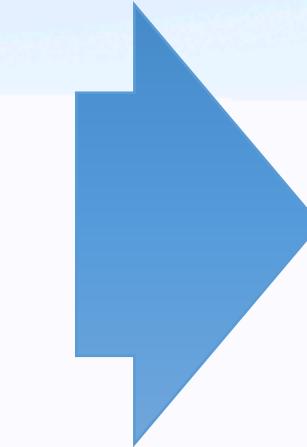
Dobrą praktyką jest utworzenie **DEDYKOWANEJ PARY KLUCZY** dedykowanej pary kluczy dla potrzeb współpracy z określonym systemem (w tym z Github)

**Jeśli wykorzystywana będzie istniejąca para kluczy to należy pamiętać o wymaganiach Github**

Note: GitHub improved security by dropping older, insecure key types on March 15, 2022.

As of that date, DSA keys ( `ssh-dss` ) are no longer supported. You cannot add new DSA keys to your personal account on GitHub.com.

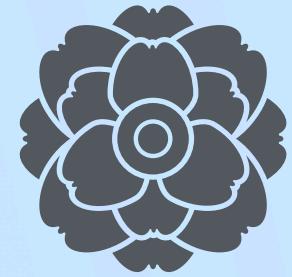
RSA keys ( `ssh-rsa` ) with a `valid_after` before November 2, 2021 may continue to use any signature algorithm. RSA keys generated after that date must use a SHA-2 signature algorithm. Some older clients may need to be upgraded in order to use SHA-2 signatures.



<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/checking-for-existing-ssh-keys>

Proces generowania kluczy i konfiguracji agenta SSH:

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>



# PAwChO – Laboratorium 6

## Github - generacja kluczy



Przykład dla Ubuntu 22.04

```
q~> ssh-keygen -t ed25519 -C "spg@spg51.net"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/slawek/.ssh/id_ed25519): /home/slawek/.ssh/gh_cli_umac_ed25519
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/slawek/.ssh/gh_cli_umac_ed25519
Your public key has been saved in /home/slawek/.ssh/gh_cli_umac_ed25519.pub
The key fingerprint is:
SHA256:usG6iAIJFPj7E4veBh703JulMEqeYJFDFwNXXX9WJaQ spg@spg51.net
The key's randomart image is:
+--[ED25519 256]--+
|..o+o... .o.o|
|.o... . . ...
|o.o .Eo
|.+o o
|..o+ . S
|+= *...
|o= B ==+
|..*o=.+o
|o.o+.+
+---[SHA256]-----+
q~>
> ls -al ~/.ssh
total 32
drwx---- 2 slawek slawek 4096 May  5 17:14 .
drwxr-x--- 25 slawek slawek 4096 May  5 16:27 ..
-rw----- 1 slawek slawek   399 May  5 17:14 gh_cli_umac_ed25519
-rw-r--r-- 1 slawek slawek   95 May  5 17:14 gh_cli_umac_ed25519.pub
-rw-rw-r-- 1 slawek slawek   41 May  5 00:17 gh_p.txt
-rw----- 1 slawek slawek 2622 May  4 23:43 id_rsa
-rw-r--r-- 1 slawek slawek  587 May  4 23:43 id_rsa.pub
-rw-r--r-- 1 slawek slawek  142 May  5 16:14 known_hosts
```

Polecenie wykorzystujące algorytm EC (patrz: zalecenia Github)

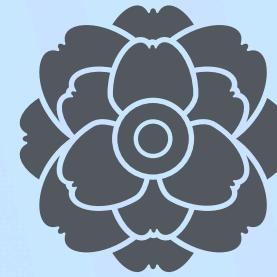
W przykładach wykorzystywany będą klucze bez passphrase

Para wygenerowanych kluczy z indywidualna nazwą (dla Github)

Rejestracja w agencie SSH

```
q~> eval "$(ssh-agent -s)"
Agent pid 16566
q~>
> ssh-add ~/.ssh/gh_cli_umac_ed25519
Identity added: /home/slawek/.ssh/gh_cli_umac_ed25519 (spg@spg51.net)
```

Proszę zmienić adres e-mail na swój adres użyty przy zakładaniu konta na Github



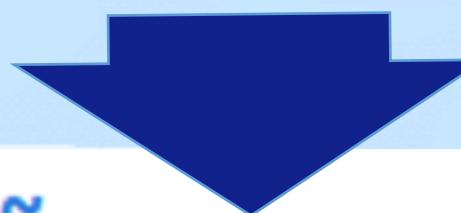
# PAwChO – Laboratorium 6

## Github - rejestracja kluczy

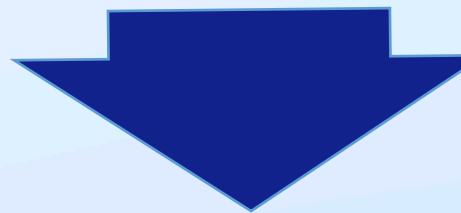
Dokumentacja procesu dodania wygenerowanych kluczy do konta na Github

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

Postępując zgodnie z opisem, w ustawieniach konta na Github powinien zostać dodany klucz publiczny SSH

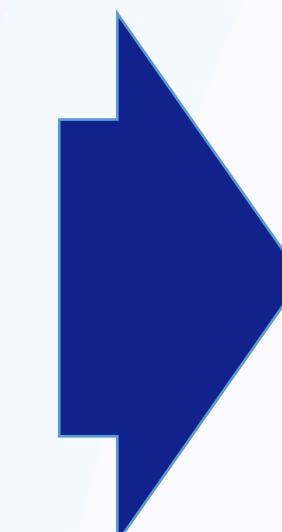


```
> cat ~/.ssh/gh_cli_umac_ed25519.pub  
ssh-ed25519 [REDACTED]
```



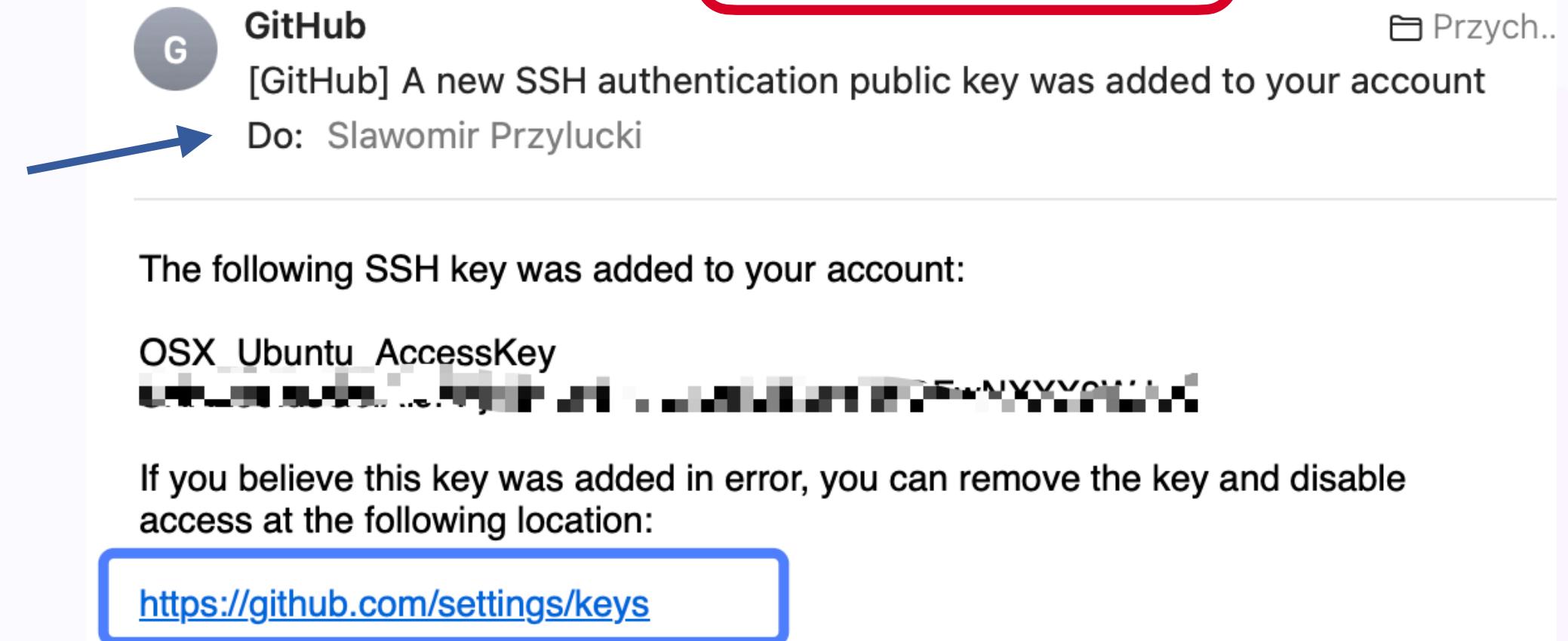
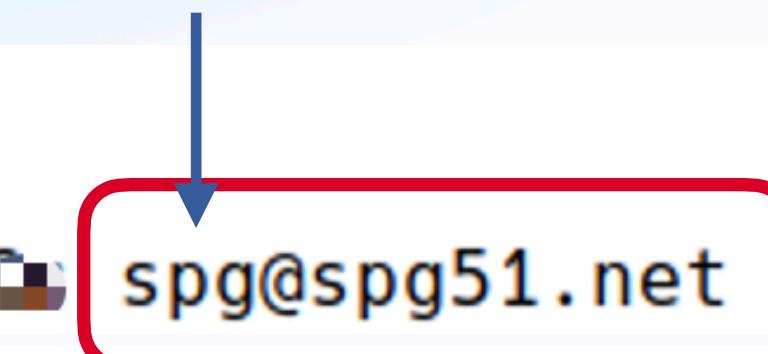
Dodanie do konta na Github

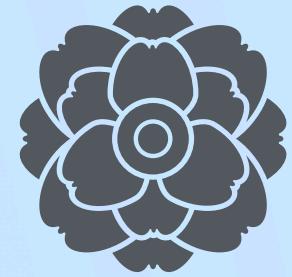
Konto Github → Settings  
→ SSH and GPG keys →  
New SSH key



Potwierdzenie dodania klucza (via e-mail)

E-mail powinien być ten sam, jaki został użyty przy rejestracji konta na Github





# PAwChO – Laboratorium 6

## GH CLI - instalacja

Instalacja w środowisko Linux (różne dystrybucje):

[https://github.com/cli/cli/blob/trunk/docs/install\\_linux.md](https://github.com/cli/cli/blob/trunk/docs/install_linux.md)



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

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



Dokumentacja i tutoriale

<https://docs.github.com/en/github-cli>



Referencyjny podręcznik Github CLI

<https://cli.github.com/manual/>

```
🍎 ~/examples/L4/Simpleimage
> gh help
Work seamlessly with GitHub from the command line.
```

### USAGE

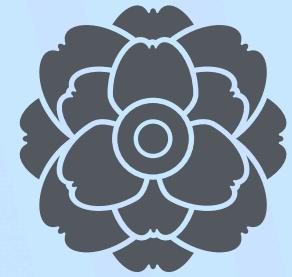
gh <command> <subcommand> [flags]

### CORE COMMANDS

auth:	Authenticate gh and git with GitHub
browse:	Open repositories, issues, pull requests, and more in the browser
codespace:	Connect to and manage codespaces
gist:	Manage gists
issue:	Manage issues
org:	Manage organizations
pr:	Manage pull requests
project:	Work with GitHub Projects.
release:	Manage releases
repo:	Manage repositories

### GITHUB ACTIONS COMMANDS

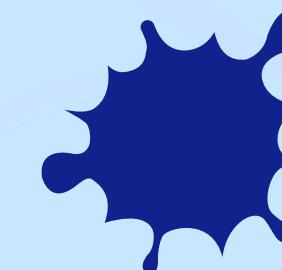
cache:	Manage GitHub Actions caches
run:	View details about workflow runs
workflow:	View details about GitHub Actions workflows



## Github - generacja PAT - cz. I

W celu korzystania z Github CLI należy utworzyć (oraz zdefiniować uprawnienia) personal-access-token

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>



Ze względu na przyszłe wykorzystywanie narzędzia Github Action, należy wygenerować token **w wersji Classic**

The screenshot shows the GitHub Developer Settings page. A blue box highlights the 'Settings / Developer settings' header. A blue arrow points down from the header to the 'Personal access tokens' section. A red arrow points from the 'Personal access tokens' section to the 'Generate new token' button. Another red arrow points from the 'Personal access tokens' section to the 'Delete' button of the token card.

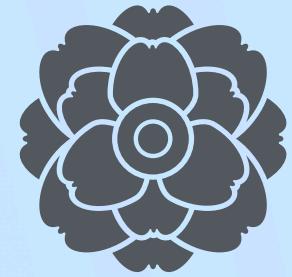
**Personal access tokens (classic)**

Tokens you have generated that can be used to access the GitHub API.

Token Name	Permissions	Last Used	Action
GENERAL_LAB_ACCESS	admin:gpg_key, admin:org, admin:public_key, admin:ssh_signing_key, codespace, delete:packages, delete_repo, project, repo, user, workflow, write:packages	Last used within the last week	Delete

⚠ This token has no expiration date.

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

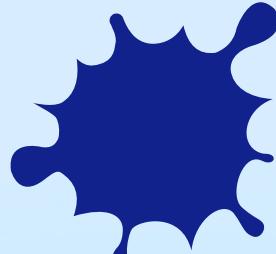


# PAwChO – Laboratorium 6

## Github - generacja PAT - cz. II

### Ustawienia (scope) - przykład

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> workflow	Update GitHub Action workflows
<input checked="" type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input checked="" type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> user	Update ALL user data
<input checked="" type="checkbox"/> read:user	Read ALL user profile data
<input checked="" type="checkbox"/> user:email	Access user email addresses (read-only)
<input checked="" type="checkbox"/> user:follow	Follow and unfollow users
<input checked="" type="checkbox"/> delete_repo	Delete repositories
<input checked="" type="checkbox"/> admin:ssh_signing_key	Full control of public user SSH signing keys
<input checked="" type="checkbox"/> write:ssh_signing_key	Write public user SSH signing keys
<input checked="" type="checkbox"/> read:ssh_signing_key	Read public user SSH signing keys



Ustawienia uprawnień dla danego tokenu można zmieniać w dowolnym momencie

Generate new token ▾

Generate new token **Beta**  
Fine-grained, repo-scoped

Generate new token (classic)  
For general use

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

GENERAL\_LAB\_ACCESS

What's this token for?

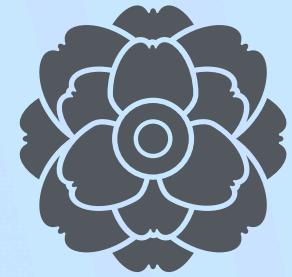
Expiration \*

No expiration The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure.  
[Learn more](#)

Dowolna nazwa (dobrze by wskazywała zakres użycia tokenu)

Dla celów zajęć - ustawiamy brak czasu wygaśnięcia tokenu



# Github - autoryzacja dostępu - cz. I

```
gh auth --help  
Authenticate gh and git with GitHub
```

```
USAGE  
  gh auth <command> [flags]
```

**AVAILABLE COMMANDS**

login:	Log in to a GitHub account
logout:	Log out of a GitHub account
refresh:	Refresh stored authentication credentials
setup-git:	Setup git with GitHub CLI
status:	View all accounts and authentication status
switch:	Switch active GitHub account
token:	Print the authentication token gh uses for a hostname and account

uses •  
**Github CLI oferuje dwie podstawowe metody logowania:**

- Autoryzacja w trybie interaktywnym
- Autoryzacja w oparciu o osobisty token dostępowy (ang. personal access token)

[https://cli.github.com/manual/gh\\_auth\\_login](https://cli.github.com/manual/gh_auth_login)

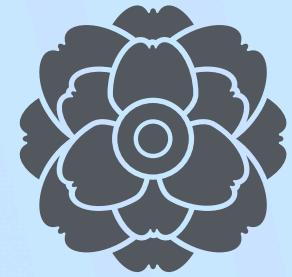
## Logowanie do Github za pomocą gh CLI:

```
gh auth login  
? Where do you use GitHub? GitHub.com  
? What is your preferred protocol for Git operations on this host? SSH  
? Upload your SSH public key to your GitHub account? /Users/slawek/.ssh/gh_osx_ed25519.pub  
? Title for your SSH key: GitHub CLI  
? How would you like to authenticate GitHub CLI? Paste an authentication token  
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens  
The minimum required scopes are 'repo', 'read:org', 'admin:public_key'.  
? Paste your authentication token: *****  
- gh config set -h github.com git_protocol ssh  
✓ Configured git protocol  
✓ SSH key already existed on your GitHub account: /Users/slawek/.ssh/gh_osx_ed25519.pub  
✓ Logged in as SenatorP51
```

**SSH preferowane dla np.  
Github Actions CI/CD**

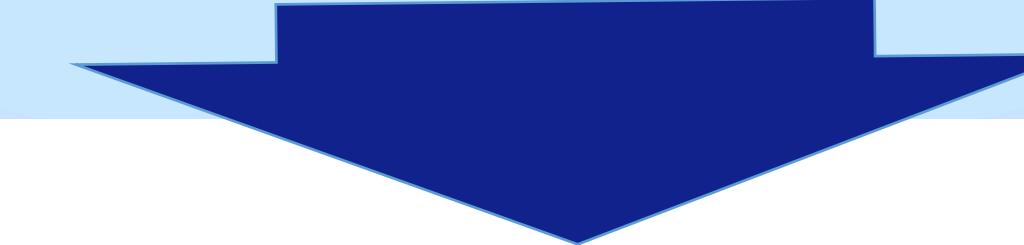
Wcześniej  
wygenerowany klucz  
publiczny SSH dla Github

Jeśli wcześniej został  
wgrany klucz publiczny  
SSH (jeśli nie, zostanie  
wgrany)

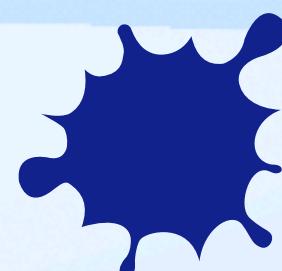


## Github - autoryzacja dostępu - cz. II

### Sprawdzenie statusu autoryzacji na GitHub

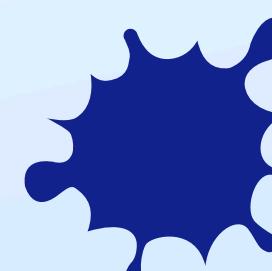


```
apple ~ > gh auth status  
github.com  
✓ Logged in to github.com account SenatorP51 (keyring)  
- Active account: true  
- Git operations protocol: ssh  
- Token: ghp_*****  
- Token scopes: 'admin:gpg_key', 'admin:org', 'admin:public_key', 'admin:ssh_signing_key', 'codespace',  
'delete:packages', 'delete_repo', 'project', 'repo', 'user', 'workflow', 'write:packages'
```



W przypadku konieczności zmiany protokołu można wykorzystać polecenie:

```
apple ~ > gh config set git_protocol ssh --host github.com
```

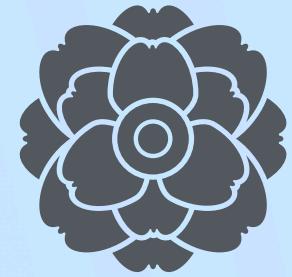


ZALECANA metoda logowanie przy użyciu tokena PAT:



```
apple ~ > gh auth login --with-token < ~/.ssh/gh_p.txt
```

Ścieżka i nazwa pliku przechowywający wygenerowany token PAT



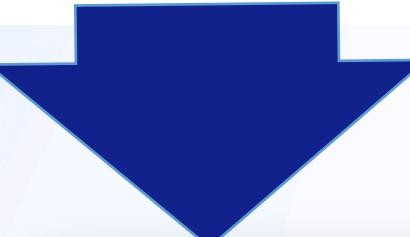
## GH CLI - zarządzanie repozytoriami na Github - cz. I

```
MacBook-Pro:~ user$ gh repo --help
Work with GitHub repositories.

USAGE
  gh repo <command> [flags]

GENERAL COMMANDS
  create:      Create a new repository
  list:        List repositories owned by user or organization

TARGETED COMMANDS
  archive:     Archive a repository
  clone:       Clone a repository locally
  delete:      Delete a repository
  deploy-key:  Manage deploy keys in a repository
  edit:        Edit repository settings
  fork:        Create a fork of a repository
  rename:      Rename a repository
  set-default: Configure default repository for this directory
  sync:        Sync a repository
  unarchive:   Unarchive a repository
  view:        View a repository
```



### ARGUMENTS

A repository can be supplied as an argument in any of the following formats:

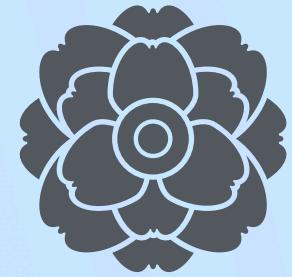
- "OWNER/REPO"
- by URL, e.g. "<https://github.com/OWNER/REPO>"

Narzędzie gh CLI pozwala na szybkie zarządzanie wszystkimi repozytoriami na Github bez konieczności korzystania z interfejsu webowego



W przypadku wykorzystania tokenu PAT, zakres wykonywanych działań musi być w ramach uprawnień przypisanych podczas generowania tego tokenu

[https://cli.github.com/manual/gh\\_repo](https://cli.github.com/manual/gh_repo)



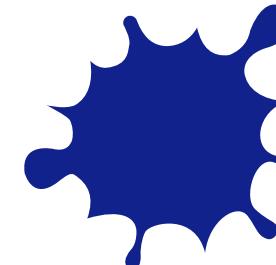
## GH CLI - zarządzanie repozytoriami na Github - cz. II

Przykład tworzenia nowego repozytorium za pomocą narzędzia gh



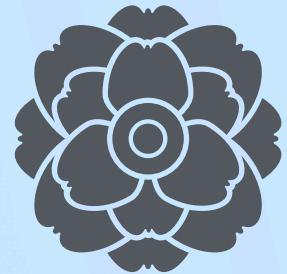
Polecenia gh można używać w trybie interaktywnym, tj. wszystkie niezbędne parametry są podawane w formie odpowiedzi na kolejne pytania lub poprzez zdefiniowanie polecenia z wymaganymi argumentami.

```
❯ gh repo create
? What would you like to do? Create a new repository on GitHub from scratch
? Repository name L6_test
? Description First repo by gh
? Visibility Public
? Would you like to add a README file? Yes
? Would you like to add a .gitignore? Yes
? Choose a .gitignore template Java
? Would you like to add a license? Yes
? Choose a license GNU General Public License v3.0
? This will create "L6 test" as a public repository on GitHub. Continue? Yes
✓ Created repository SenatorP51/L6_test on GitHub
https://github.com/SenatorP51/L6 test
? Clone the new repository locally? Yes
Klonowanie do „L6_test”...
```



Przykład poniżej wykorzystuje metodę interaktywną, dostępne argumenty są natomiast opisane w dokumentacji, np:

[https://cli.github.com/manual/gh\\_repo\\_create](https://cli.github.com/manual/gh_repo_create)



# PAwChO – Laboratorium 6

## GH CLI - zarządzanie repozytoriami na Github - cz. III

QUESTION

Ponieważ w trakcie tworzenia repo wybrana została opcja klonowania do katalogu lokalnego to otworzony został katalog wraz z konfiguracją git

```
apple ~ /examples/L6/L6_test $ main [✓]
> ll -a
drwxr-xr-x@ - slawek staff 21 lut 22:24 .git
.rw-r--r--@ 290 slawek staff 21 lut 22:24 .gitignore
.rw-r--r--@ 35k slawek staff 21 lut 22:24 LICENSE
.rw-r--r--@ 27 slawek staff 21 lut 22:24 README.md
```



CLI

```
apple ~ /examples/L6/L6_test $ main [✓]
> gh repo list | grep L6
SenatorP51/L6_test First repo by gh public
```

Potwierdzenie poprawności utworzenia repozytorium

WEB

L6\_test Public

main 1 Branch 0 Tags

Go to file Add file Code

SenatorsP51 Initial commit 902c48e · 9 minutes ago 1 Commit

.gitignore Initial commit 9 minutes ago

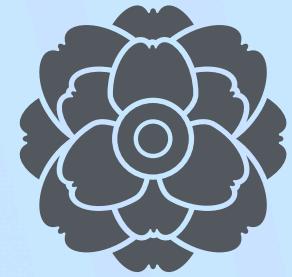
LICENSE Initial commit 9 minutes ago

README.md Initial commit 9 minutes ago

README GPL-3.0 license

L6\_test

First repo by gh



## Github - repozytorium ghcr - cz. I

Rejestr kontenerów **ghcr.io** przechowuje obrazy kontenerów dla danej organizacji lub dla danego konta osobistego i umożliwia skojarzenie obrazu z repozytorium. Repozytorium umożliwia tworzenie zasad dostępu do obrazów lub oferuje anonimowy dostęp do publicznych obrazów kontenerowych.

<https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-container-registry>



### Dwie ważne uwagi wynikające z dokumentacji

#### Authenticating to the Container registry ↗

GitHub Packages only supports authentication using a personal access token (classic). For more information, see "[Managing your personal access tokens](#)."

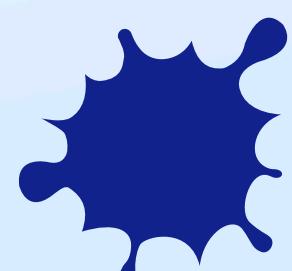


#### About Container registry support #

The Container registry currently supports the following container image formats:

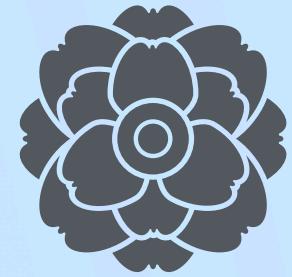
- [Docker Image Manifest V2, Schema 2](#)
- [Open Container Initiative \(OCI\) Specifications](#)

When installing or publishing a Docker image, the Container registry supports foreign layers, such as Windows images.



Repozytorium jest częścią usługi Github Packages. Proszę pamiętać o ograniczeniach dla darmowych kont na GitHub.

<https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-container-registry#troubleshooting>



## Github - repozytorium ghcr - cz. II

Logowanie się na repozytorium [ghcr.io](https://ghcr.io) (UWAGA: należy uruchomi Docker Desktop)

```
~ export CR_LAB=...  
~ echo $CR_LAB | docker login ghcr.io -u SenatorP51 --password-stdin  
Login Succeeded
```

Zmienna przechowująca  
osobisty token dostępowy

!!!!!!Proszę wstawić nazwę  
swojego konta na Github!!!!!



**W nazwach obrazów  
przesyłanych na [ghcr.io](https://ghcr.io)  
NIE MOŻNA używać  
dużych liter**

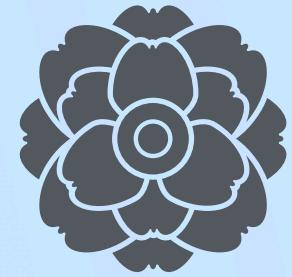
```
~ docker tag alpine ghcr.io/senatorp51/myalpine:test  
~ docker push ghcr.io/senatorp51/myalpine:test  
The push refers to repository [ghcr.io/senatorp51/myalpine]  
6e771e15690e: Pushed  
test: digest: sha256:757d680068d77be46fd1ea20fb21db16f150468c5e7079a08a2e4705aec096ac size: 1025
```

!!!!!! Proszę wstawić nazwę  
swojego konta na Github !!!!!!!



**Właściwe operowanie obrazem w ramach repozytorium ghrc ale również w odniesieniu do  
usług skojarzonych, np. GHActions wymaga stosowania właściwych etykiet oraz adnotacji**

<https://docs.github.com/en/packages/working-with-a-github-packages-registry/working-with-the-container-registry#labelling-container-images>



# PAwChO – Laboratorium 6

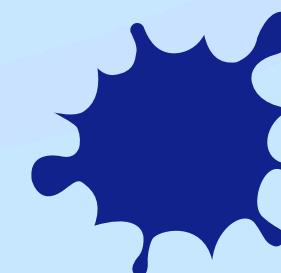
## Github - repozytorium ghcr - cz. III

Dostęp do wgranego obrazu poprzez interfejs webowy Github

The screenshot shows the GitHub user interface for packages. At the top, there's a navigation bar with 'Overview', 'Repositories 18', 'Projects', 'Packages 5', and 'Stars'. A blue arrow points from the 'Packages' button to the 'myalpine test' package card. Another blue arrow points from the 'Private' status indicator on the package card to the text 'Domyślnie, wgrany obraz ma atrybut „private” - można go zmienić na „public” !!!!'.

**Domyślnie, wgrany obraz ma atrybut „private” - można go zmienić na „public” !!!!**

Domyślnie, wgrany obraz ma atrybut „private” - można go zmienić na „public” !!!!



MOŻLIWE jest połączenie repo ghcr (obrazów) z repozytorium git

The screenshot shows the 'Link this package to a repository' section. It includes a 'Connect Repository' button, a note about linking to a repository, and a 'Dockerfile' section with instructions and examples.

**Link this package to a repository**

By linking to a repository, you can automatically add a Readme, link discussions, and show contributors on this page.

**Connect Repository**

Or link via Dockerfile

To locally connect your container image to a repository, you must add this line to your Dockerfile:

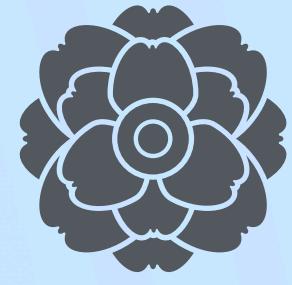
```
LABEL org.opencontainers.image.source https://github.com/OWNER/REPO
```

Note: To connect a repository to your container image, the namespace for the repository and container image on GitHub must be the same. For example, they should be owned by the same user or organization.



Proszę KONIECZNIE zapoznać się ze sposobami deklarowania uprawnień dostępu do obrazów

<https://docs.github.com/en/packages/learn-github-packages/configuring-a-packages-access-control-and-visibility>



## ghcr.io - rola adnotacji obrazów

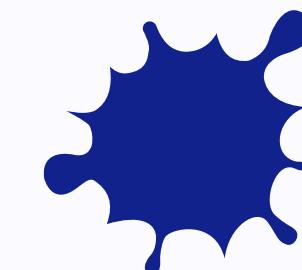


Etykiety nie mają znaczenia dla samego procesu budowania obrazów czy też działanie kontenerów. Są one wykorzystywane przez zewnętrzne narzędzia zarządzające obrazami/kontenerami.

Pierwotnie Docker wykorzystywał swój własny schemat etykietowania obrazów, nazywany *Label Schema*. Obecnie NALEŻY stosować schemat zawarty w specyfikacji obrazów, zdefiniowany przez OCI (ang. Open Containers Initiative)

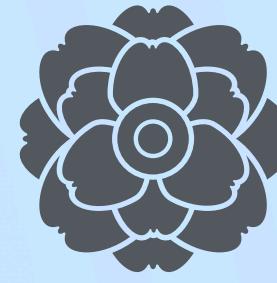
W przypadku korzystania z ghcr.io, szczególnie zalecane jest wykorzystanie :

- `org.opencontainers.image.source`
- `org.opencontainers.image.description`
- `org.opencontainers.image.authors`



Pełna lista adnotacji OCI jak i ich odpowiedniki w starym standardzie *Label Schema*:

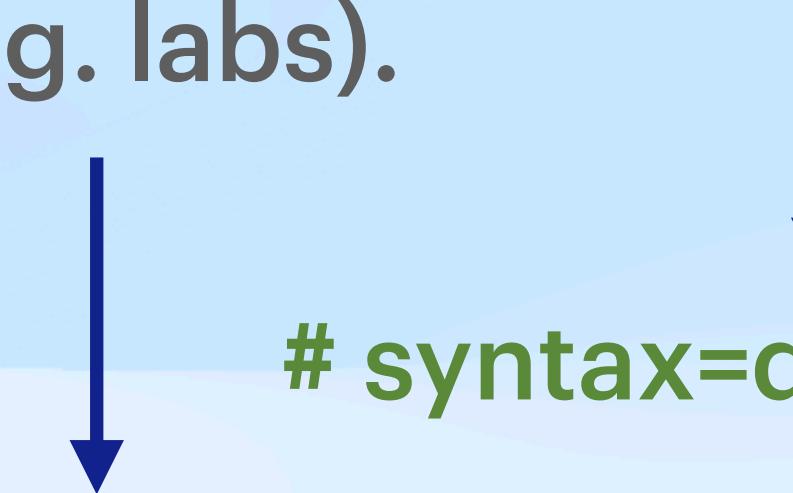
<https://github.com/opencontainers/image-spec/blob/main/annotations.md>



# PAwChO – Laboratorium 6

## Rozszerzony frontend w buildx

Docker dystrybuuje oficjalne wersje obrazów dla rozszerzonych frontend-ów w Docker Hub. Istnieją dwa kanały, w których publikowane są nowe obrazy: stabilny (ang. stable) i rozwojowy (ang. labs).

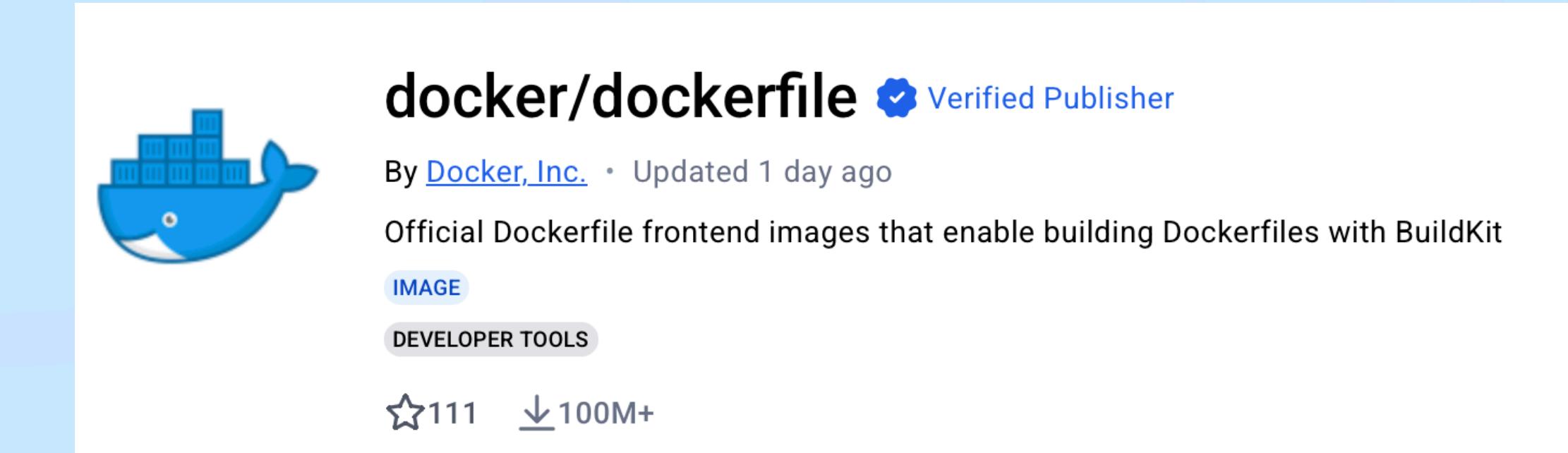


# syntax=docker/dockerfile:1.2-labs

<https://docs.docker.com/build/buildkit/dockerfile-frontend/#official-releases>



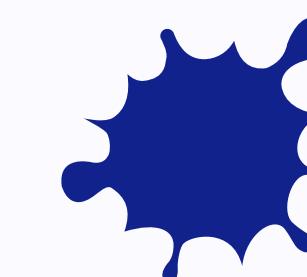
Rozszerzony frontend jest obecnie jedyną metodą na bezpieczne wykorzystanie danych wrażliwych (ang. Secrets) w procesie budowania obrazów.



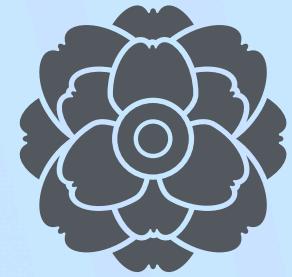
<https://hub.docker.com/r/docker/dockerfile>

Składnia odwołania się do tego obrazu:

# syntax=[remote image reference]



Powyższe odwołania MUSI być pierwszą linią w pliku Dockerfile



## Rozszerzony frontend - secrets - przykład - cz. I

1

Utworzenie dockerfile wykorzystującego wymaganą funkcjonalność rozszerzonego frontendu - w przykładzie: mount secrets

```
~/examples/L6
> cat .hidden.txt
Bardzo tajne dane
```



```
Dockerfile_s x
Dockerfile_s > ...
1 # syntax=docker/dockerfile:1.2
2
3 FROM ubuntu
4 RUN --mount=type=secret,id=mysecret,dst=/dark.txt
5 CMD [ "/bin/bash" ]
```

Wersje obrazu rozszerzonego frontendu należy zweryfikować w dokumentacji, np. dla tego przykładu:

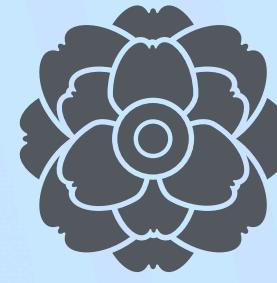
<https://docs.docker.com/reference/dockerfile/#run---mounttypesecret>

2

Budowa obrazu ze wskazaniem źródła dla secret-u o id mysecret

```
~/examples/L6
> docker build --secret id=mysecret,src=.hidden.txt -f Dockerfile_s -t local/secret:v1 .
[+] Building 3.5s (11/11) FINISHED
=> [internal] load build definition from Dockerfile_s
=> => transferring dockerfile: 199B
=> resolve image config for docker-image://docker.io/docker/dockerfile:1.2
=> [auth] docker/dockerfile:pull token for registry-1.docker.io
=> docker-image://docker.io/docker/dockerfile:1.2@sha256:e2a8561e419ab1ba6b2fe6cbdf49fd92b95912df1cf7d313c3e2230a333fdbcc
=> => resolve docker.io/docker/dockerfile:1.2@sha256:e2a8561e419ab1ba6b2fe6cbdf49fd92b95912df1cf7d313c3e2230a333fdbcc
=> => sha256:2904710679e8cf1b7e70b02e78dc381d89d57be88fa9add57034e56b396c220a 9.01MB / 9.01MB
=> => extracting sha256:2904710679e8cf1b7e70b02e78dc381d89d57be88fa9add57034e56b396c220a
=> [internal] load build definition from Dockerfile_s
=> [internal] load .dockerignore
```

Pobranie i użycie obrazu frontend-u w wersji 1.2



## Rozszerzony frontend - secrets - przykład - cz. II

3

Weryfikacja czy wrażliwe dane są/nie są w jakikolwiek sposób widoczne w strukturze obrazu

```
~/examples/L6
> docker history local/secret:v1
IMAGE          CREATED      CREATED BY
f6843ba735c7  44 minutes ago  CMD ["/bin/bash"]
<missing>     44 minutes ago  RUN /bin/sh -c # buildkit
<missing>     3 weeks ago    /bin/sh -c #(nop)  CMD ["/bin/bash"]
<missing>     3 weeks ago    /bin/sh -c #(nop) ADD file:68158f1ff76fd4de9...
<missing>     3 weeks ago    /bin/sh -c #(nop) LABEL org.opencontainers...
<missing>     3 weeks ago    /bin/sh -c #(nop) LABEL org.opencontainers...
<missing>     3 weeks ago    /bin/sh -c #(nop) ARG LAUNCHPAD_BUILD_ARCH
<missing>     3 weeks ago    /bin/sh -c #(nop) ARG RELEASE
0B             4.1kB        0B           buildkit.dockerfile.v0
0B             110MB        0B           buildkit.dockerfile.v0
0B             0B            0B           buildkit.dockerfile.v0
```

4

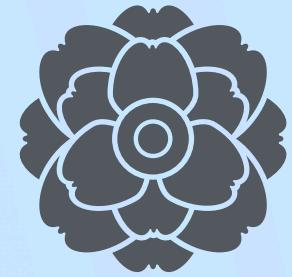
Weryfikacja czy wrażliwe dane są/nie są dostępne w kontenerze uruchomionym na bazie zbudowanego obrazu

```
~/examples/L6
> docker run -it --rm --name tajniak local/secret:v1 /bin/bash
root@156b91f47036:/# cat dark.txt
cat: dark.txt: No such file or directory
root@156b91f47036:/#
```

Czy taki wynik działania to sukces czy ... porażka

??????



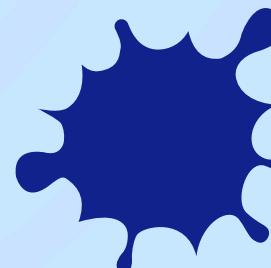


## Rozszerzony frontend a kontekst procesu build

```
~/examples/L6
> docker build --help
Start a build

Usage: docker buildx build [OPTIONS] PATH | URL | -
```

Jako URL można podać adres repozytorium git i w ten sposób wykorzystać je jako kontekst do budowy obrazu



W przypadku repo git jako kontekstu, Builder kopioje domyślną gałąź kodu i pobiera tylko ostatni commit HEAD, a nie całą historię. Takie domyślne zachowanie można zmodyfikować poprzez zastosowanie tzw. Fragmentacji URL (ang. URL fragments)

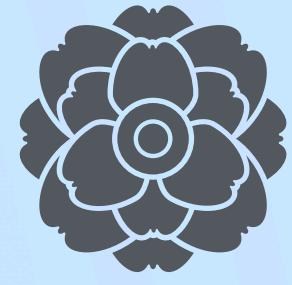
<https://docs.docker.com/build/concepts/context/#git-repositories>

Jeśli kontekst Git wskazuje na prywatne repozytorium, proces build wymaga dostarczenia danych uwierzytelniających. Można użyć uwierzytelniania opartego na SSH lub tokenie PAT.

<https://docs.docker.com/build/concepts/context/#private-repositories>



Jeżeli zdalny kontekst (np. repo git) wymaga uwierzytelniania to należy używać opcji **--ssh** oraz **--secret** w poleceniu `docker build ...`. NIE WOLNO w tym celu używać opcji `--build-arg`

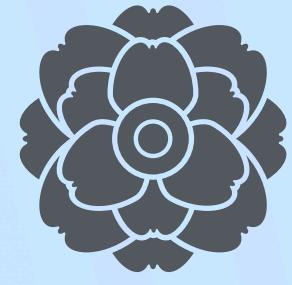


## Zadanie do wykonania - NIEOBOWIĄZKOWE - cz. I

Zgodnie z instrukcją do laboratorium 5, jako zadanie obowiązkowe należało opracować plik Dockerfile dla aplikacji webowej uruchamianej w oparciu o serwer Nginx oraz budowanej w oparciu o metodę wieloetapową. W tym zadaniu nieobowiązkowym opracowana aplikacja zostanie wykorzystana rozwiążanie tego zadania.

**Zadanie polega na:**

- Wykorzystaniu klienta CLI systemy Github w postaci programu gh w taki sposób, by utworzyć repozytorium publiczne na Github o nazwie **pawcho6** i powiązać go z katalogiem zawierającym rozwiązanie zadania obowiązkowego z lab. 5.
- Zmodyfikowaniu istniejącego pliku Dockerfile z rozwiązania z lab. 5 tak by pełnił on rolę frontendu dla silnika Buildkit
- Zmodyfikowaniu pliku Dockerfile tak by pozwalał pobrać za pomocą odpowiednich instrukcji wewnętrz Dockerfile zawartość przygotowanego repozytorium **pawcho6** i na jego podstawie zbudowanie obraz Docker. Pobieranie ma być realizowane za pomocą protokołu SSH



## Zadanie do wykonania - NIEOBOWIĄZKOWE - cz. II

### Zadanie polega na - cd:

- W procesie budowania nadaniu obrazowi tag-u *lab6* i przesłaniu go do swojego repozytorium na Github (repo: ghcr.io).
- Zamianie atrybutu dostępu do zbudowanego obrazu na repo na Github z private na public.
- Powiązaniu tego repozytorium obrazów z utworzonym na początku zadania, repozytorium git o nazwie *pawcho6*

**W sprawozdaniu należy:**

- Podać treść utworzonego pliku Dockerfile
- Podać użyte polecenie do budowy obrazu i wynik jego działania
- Potwierdzenie wykonania pozostałych zadań cząstkowych

### Materiały pomocnicze

<https://docs.docker.com/build/building/secrets/>

<https://docs.docker.com/engine/reference/builder/#run---mounttypessh>

<https://medium.com/@tonistiigi/build-secrets-and-ssh-forwarding-in-docker-18-09-ae8161d066>