

POLITECHNIKA  
LUBELSKA  
WYDZIAŁ ELEKTROTECHNIKI  
I INFORMATYKI

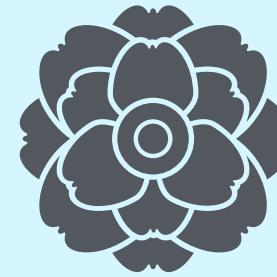
# Programowanie Aplikacji w Chmurze Obliczeniowej

## Laboratorium nr 9

Konfiguracja i uruchomienie przykładowego łańcucha CI w oparciu o Github oraz usługę Github Actions. Automatyzacja budowania obrazów z wykorzystaniem buldkitd oraz z własnym schematem tagowania

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





## Przygotowanie środowiska CI - cz. I

1

Przed rozpoczęciem działań opisanych w tej instrukcji należy zagwarantować, że:

- w wykorzystywanym systemie jest zainstalowane (i skonfigurowane) narzędzie git,
- w wykorzystywanym systemie jest zainstalowane (i skonfigurowane) narzędzie gh (lab 6)
- student zalogowany jest w systemie GitHub z użyciem dedykowanego tokenu PAT (lab 6)



```
mac ~ /Labs/lab9
> 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'
```

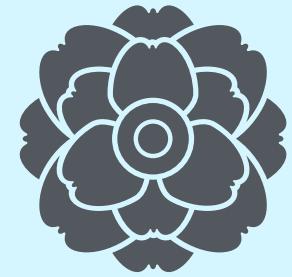
```
uses 🛠 desktop-linux
```

2

Do wybranego podkatalogu w katalogu domowym należy rozpakować pliki źródłowe dostępne na moodle, w katalogi do laboratorium 9. Plik: lab9\_examples.zip



```
mac ~ /Labs/lab9
> ll -a
drwxr-xr-x@ 10 slawek staff 22 mar 21:26 .github
.rw-r--r--@ 263 slawek staff 23 mar 19:26 git.gitignore
.rw-r--r--@ 11k slawek staff 22 mar 21:26 Dockerfile
.rw-r--r--@ 286 slawek staff 23 mar 20:34 LICENSE
drwxr-xr-x@ - slawek staff 22 mar 21:26 README.md
drwxr-xr-x@ - slawek staff 22 mar 21:26 src
```



## Przygotowanie środowiska CI - cz. II

3

Należy zainicjować repo git

```
apple ~ ~/Labs/lab9
> git init -b main
Zainicjowano puste repozytorium Gita w /Users/slawek/Labs/lab9/.git/
apple ~ ~/Labs/lab9 ⏺ main [!]
```

4

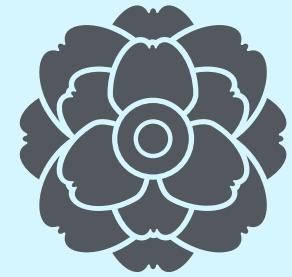
Na Github należy utworzyć repozytorium na podstawie repo z punktu 3 (klonowanie)

```
apple ~ ~/Labs/lab9 ⏺ main [!]
> gh repo create
? What would you like to do? Push an existing local repository to GitHub
? Path to local repository .
? Repository name lab9
? Description Repozytorium dedykowane zadaniom z laboratorium 9
? Visibility Public
✓ Created repository SenatorP51/lab9 on GitHub
https://github.com/SenatorP51/lab9
? Add a remote? Yes
? What should the new remote be called? origin
✓ Added remote git@github.com:SenatorP51/lab9.git
```

5

Sprawdzenie utworzenia repo na Github

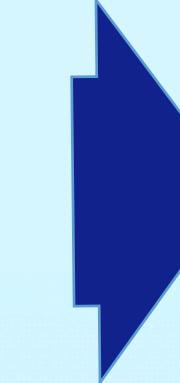
```
apple ~ ~/Labs/lab9 ⏺ main [!]
> gh repo list | grep lab9
SenatorP51/lab9 Repozytorium dedykowane zadaniom z laboratorium 9      public
```



## Przygotowanie środowiska CI - cz. III

6

Przygotowanie do pierwszego commit-a:



```
 MacBook ~ /Labs/lab9 ⏺ main [!] 
> git add .
MacBook ~ /Labs/lab9 ⏺ main [++(6)]
> git status
Na gałęzi main
```

Jeszcze nie ma zapisów

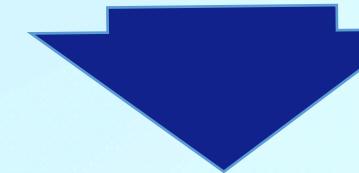
Zmiany do złożenia:

(użyj „git rm --cached <plik>...”, aby wycofać)

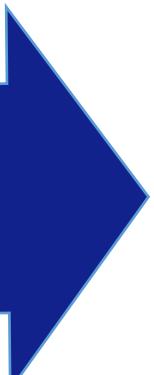
nowy plik:	.github/workflows/gha_example.yml
nowy plik:	.gitignore
nowy plik:	Dockerfile
nowy plik:	LICENSE
nowy plik:	README.md
nowy plik:	src/index.html

7

Wykonanie pierwszego commit-a:



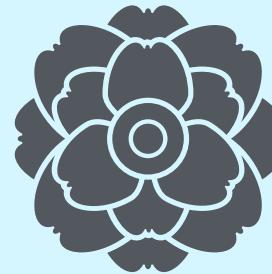
```
 MacBook ~ /Labs/lab9 ⏺ main [++(6)]
> git commit -m "Inicjalizacja repo lab9"
[main (zapis-korzeń) cf37a2e] Inicjalizacja repo lab9
  6 files changed, 295 insertions(+)
  create mode 100755 .github/workflows/gha_example.yml
  create mode 100644 .gitignore
  create mode 100644 Dockerfile
  create mode 100644 LICENSE
  create mode 100644 README.md
  create mode 100644 src/index.html
```



```
 MacBook ~ /Labs/lab9 ⏺ main
> git push -u origin main
Wymienianie obiektów: 11, gotowe.
Zliczanie obiektów: 100% (11/11), gotowe.
Kompresja delt z użyciem do 10 wątków
Kompresowanie obiektów: 100% (7/7), gotowe.
Zapisywanie obiektów: 100% (11/11), 5.54 KiB | 5.54 MiB/s, gotowe.
Total 11 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:SenatorP51/lab9.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

8

Synchronizacja z repo na Github:



# PAwChO – Laboratorium 9

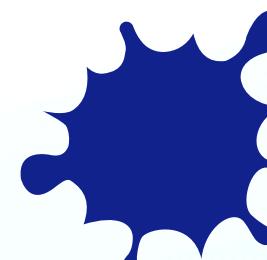
## Przygotowanie środowiska CI – cz. IV

9

Należy sprawdzić czy na Github zostało poprawnie utworzone repozytorium *lab 9*

10

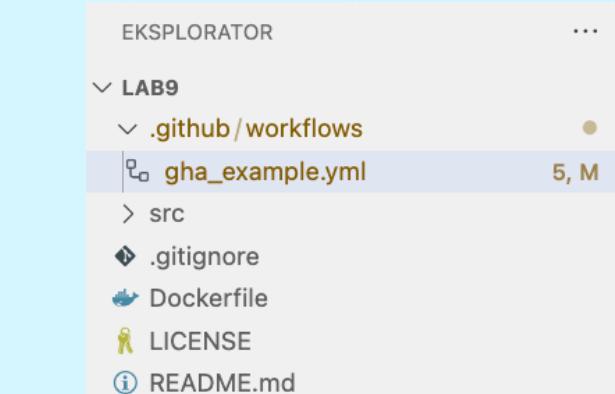
W ramach udostępnionych źródeł dostępny jest przykładowy łańcuch CI dla usługi GitHub Actions



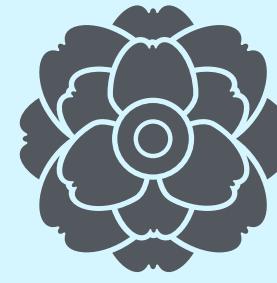
Przydatne linki do analizy i samodzielnego tworzenia łańcuchów CI/CD w oparciu o Github Actions

<https://docs.github.com/en/actions/quickstart>

<https://github.com/actions/starter-workflows>



```
gha_example.yml 5, M ×
.github > workflows > gha_example.yml
1 name: GHAction example
2
3 on:
4   workflow_dispatch:
5     push:
6       tags:
7         - 'v*'
8
9 jobs:
10 ci_step:
11   name: Build, tag and push Docker image to DockerHub
12   runs-on: ubuntu-latest
13
14 steps:
15
16   - name: Check out the source_repo
17     uses: actions/checkout@v4
18
19
20   - name: Docker metadata definitions
21     id: meta
22     uses: docker/metadata-action@v5
23     with:
24       images: ${{ vars.DOCKERHUB_USERNAME }}/example
25       flavor: latest=false
26       tags: |
27         type=sha,priority=100,prefix=sha-,format=short
28         type=semver,priority=200,pattern={{version}}
29
30
31   - name: QEMU set-up
32     uses: docker/setup-qemu-action@v3
33
34
35   - name: Buildx set-up
36     uses: docker/setup-buildx-action@v3
37
38
39   - name: Login to DockerHub
40     uses: docker/login-action@v3
41     with:
42       username: ${{ vars.DOCKERHUB_USERNAME }}
43       password: ${{ secrets.DOCKERHUB_TOKEN }}
44
45
46   - name: Build and push Docker image
47     uses: docker/build-push-action@v5
48     with:
49       context: .
50       file: ./Dockerfile
51       platforms: linux/amd64,linux/arm64
52       push: true
53       cache-from: |
54         type=registry,ref=${{ vars.DOCKERHUB_USERNAME }}/example:cache
55       cache-to: |
56         type=registry,ref=${{ vars.DOCKERHUB_USERNAME }}/example:cache
57
58       tags: ${{ steps.meta.outputs.tags }}
```



# PAwChO – Laboratorium 9

## Łańcuch CI - Checkout

Ten moduł *Action* sprawdza dane repozytorium zdefiniowane przez zmienną `$GITHUB_WORKSPACE`, aby potwierdzić że uruchamiany łańcuch może uzyskać do niego dostęp na zasadach określonych w konfiguracji modułu.

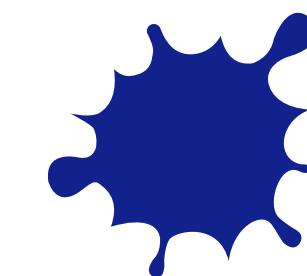
### Scenarios

- [Fetch only the root files](#)
- [Fetch only the root files and .github and src folder](#)
- [Fetch only a single file](#)
- [Fetch all history for all tags and branches](#)
- [Checkout a different branch](#)
- [Checkout HEAD<sup>^</sup>](#)
- [Checkout multiple repos \(side by side\)](#)
- [Checkout multiple repos \(nested\)](#)
- [Checkout multiple repos \(private\)](#)
- [Checkout pull request HEAD commit instead of merge commit](#)
- [Checkout pull request on closed event](#)
- [Push a commit using the built-in token](#)

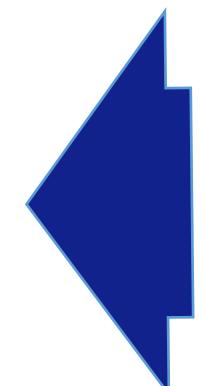
GitHub Action  
Checkout  
v4.1.4 Latest version  
Build and Test passing

### Checkout V4

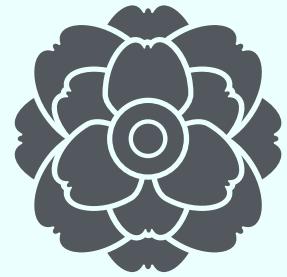
<https://github.com/marketplace/actions/checkout>



W konfiguracji można określić jak zamierza się korzystać z repo - tzw. **Dfefinicja scenariuszy**



<https://github.com/marketplace/actions/checkout#scenarios>



# PAwChO – Laboratorium 9

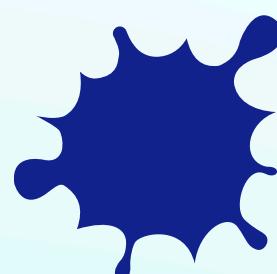
## Łańcuch CI - Logowanie - cz. I

A screenshot of the GitHub Marketplace page for the "Docker Login" action. The action is listed under "GitHub Action". It has a logo of a Docker container, the name "Docker Login", version "v3.1.0", and a "Latest version" button. Below the main listing, there are status indicators: "release v3.1.0", "marketplace docker-login", "ci passing", "test passing", and "coverage 72%".

<https://github.com/marketplace/actions/docker-login>

```
name: Login to DockerHub  
uses: docker/login-action@v3  
with:  
username: ${{ vars.DOCKERHUB_USERNAME }}  
password: ${{ secrets.DOCKERHUB_TOKEN }}
```

Moduł Action docker/login-action@v3 pozwala na zalogowanie się w wielu repozytoriach obrazów.  
PROSZĘ KONIECZNIE zapoznać się z dokumentacją



### Docker Hub - generacja tokenu dostępowego

1

A screenshot of the Docker Hub account settings. The "Account Settings / Security" section is highlighted with a red box. A large blue arrow points down to the "Access Tokens" section.

2

A screenshot of the Docker Hub access tokens page. The "Access Tokens" section is highlighted with a red box. A green arrow points from the previous screenshot to this one, indicating the next step.

3

#### New Access Token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more](#)

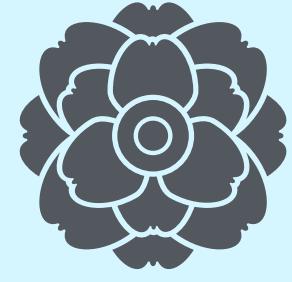
A screenshot of the "New Access Token" generation form. The "Access Token Description \*" field contains "GA\_ACCESS\_TOKEN\_L9" and is highlighted with a green box. The "Access permissions" dropdown is set to "Read, Write, Delete" and is highlighted with a blue box. A red arrow points from the "Generate" button to the "Access permissions" dropdown. A blue arrow points up from the "Access Tokens" section in the previous screenshot to this form.

Cancel

Generate

Read, Write, Delete tokens allow you to manage your repositories.

New Access Token



# PAwChO – Laboratorium 9

## Łańcuch CI - Logowanie - cz. II



4

### Copy Access Token

When logging in from your Docker CLI client, use this token as a password. [Learn more](#)

ACCESS TOKEN DESCRIPTION

GA\_ACCESS\_TOKEN\_L9

ACCESS PERMISSIONS

Read, Write, Delete

To use the access token from your Docker CLI client:

1. Run

```
docker login -u spg51
```

2. At the password prompt, enter the personal access token.

```
[REDACTED]
```

**WARNING:** This access token will only be displayed once. It will not be stored and cannot be retrieved. Please be sure to save it now.

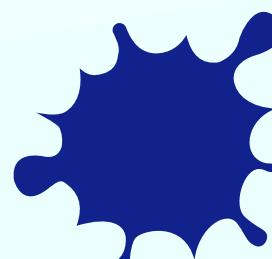
**Copy and Close**

**UWAGA: !!!!!!! Podobnie jak z PAT na GitHub (lab 6) należy zapisać wygenerowany token w bezpiecznym miejscu (np. plik w katalogu `~/.ssh`) oraz zapewnić, że nie pozostanie on na komputerze uczelnianym po zakończeniu zajęć !!!!!!**

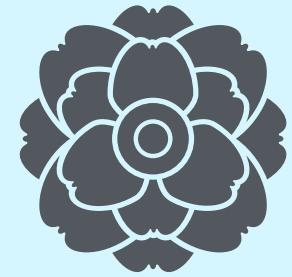
5

**Należy w utworzonym repo na Github zdefiniować odpowiednio: zmieną oraz sekret**

DOCKERHUB\_USERNAME  
DOCKERHUB\_TOKEN



**Zasięg wygenerowanego tokena dostępowego odnosi się do wszystkich repozytoriów (całego konta na DockerHub)**



# PAwChO – Laboratorium 9

## Łańcuch CI - Logowanie - cz. III

Działania na utworzonym repo *lab9*

1A

```
apple ~ ~/Labs/lab9 🏙 main [✓]
> gh secret set DOCKERHUB_TOKEN
? Paste your secret: *****
✓ Set Actions secret DOCKERHUB_TOKEN for SenatorP51/lab9
```

Należy wkleić utworzony na  
DockerHub token dostępowy PAT

1B

### Weryfikacja utworzenia secret-u w interfejsie web Github

The screenshot shows the 'Repository secrets' page for a repository. On the left, there's a sidebar with links like 'Security', 'Code security and analysis', 'Deploy keys', 'Secrets and variables' (which is highlighted with a green border), and 'Actions'. The main area shows a table with one row for a secret named 'DOCKERHUB\_TOKEN'. A red arrow points from the 'Secrets and variables' link in the sidebar to the 'DOCKERHUB\_TOKEN' entry in the table.

2A

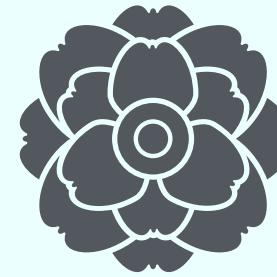
```
apple ~ ~/Labs/lab9 🏙 main [✓]
> gh variable set DOCKERHUB_USERNAME
? Paste your variable spg51
✓ Created variable DOCKERHUB_USERNAME for SenatorP51/lab9
```

Należy podać własną nazwę  
użytkownika na Dockerhub

2B

### Weryfikacja utworzenia zmiennej w interfejsie web Github

The screenshot shows the 'Repository variables' page for a repository. On the left, there's a sidebar with links like 'Security', 'Code security and analysis', 'Deploy keys', 'Secrets and variables' (which is highlighted with a green border), and 'Actions'. The main area shows a table with one row for a variable named 'DOCKERHUB\_USERNAME' with the value 'spg51'. A red arrow points from the 'Secrets and variables' link in the sidebar to the 'DOCKERHUB\_USERNAME' entry in the table.



PAwChO – Laboratorium 9

## Łańcuch CI - Deklarowanie środowiska budowanie obrazów

Marketplace / Actions / Docker Setup Buildx

 GitHub Action  
**Docker Setup Buildx**  
v3.0.0 [Latest version](#)

release v3.0.0 | marketplace docker-setup-buildx | ci passing | test passing | coverage 40%

**About**  
GitHub Action to set up Docker [Buildx](#).

<https://github.com/marketplace/actions/docker-setup-buildx>

Marketplace / Actions / Docker Setup QEMU

 GitHub Action  
**Docker Setup QEMU**  
v3.0.0 [Latest version](#)

release v3.0.0 | marketplace docker-setup-qemu | ci passing | test passing | coverage 100%

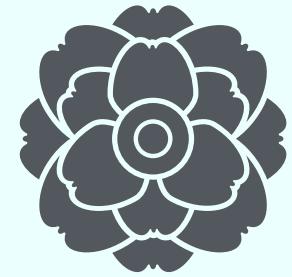
**About**  
GitHub Action to install [QEMU](#) static binaries.

<https://github.com/marketplace/actions/docker-setup-qemu>

**Powyższe Actions są zwykle stosowane łącznie. Pozwalają na danym hoście runner-a zainstalować wszystkie niezbędne komponenty do budowania obrazu z wykorzystaniem silnika Buildkit.**



Bez instalacji tych składników nie możliwe jest budowanie obrazów na wiele architektur sprzętowych ani wykorzystanie rozproszonego cache-a.



# PAwChO – Laboratorium 9

## Łańcuch CI - Tagowanie obrazów - cz. I

Marketplace / Actions / Docker Metadata action

 GitHub Action

**Docker Metadata action**

v5.5.1 Latest version

release v5.5.1 marketplace docker-metadata-action ci passing test passing coverage 78%

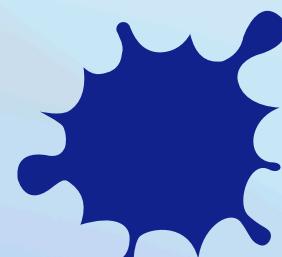
**About**

GitHub Action to extract metadata from Git reference and GitHub events. This action is particularly useful if used with [Docker Build Push](#) action to tag and label Docker images.

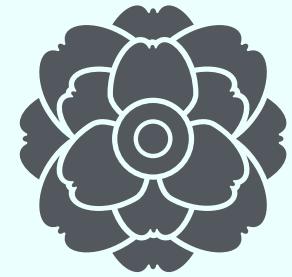
```
name: Docker metadata definitions
id: meta
uses: docker/metadata-action@v5
with:
  images: ${{ vars.DOCKERHUB_USERNAME }}/example2
  flavor: latest=false
  tags:
    type=sha,priority=100,prefix=sha-,format=short
    type=semver,priority=200,pattern={{version}}
```

<https://github.com/marketplace/actions/docker-metadata-action>

**Deklaracja nazwy obrazu (bez tag-u):** docker.io/<dockerhub\_user>/<user\_repo\_name>



Id: <nazwa> nie jest bezpośrednio związana z konfiguracją tego modułu Actions. Jest to metoda przypisywania nazwy do danego kroku (danego step-u) aby można było w prosty sposób odwoływać się do niego w innych miejscach łańcucha CI



# PAwChO – Laboratorium 9

## Łańcuch CI - Tagowanie obrazów - cz. II

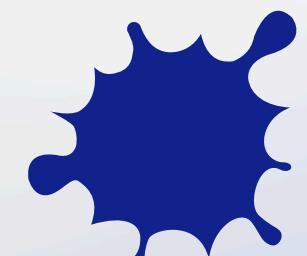
```
-  
  name: Docker metadata definitions  
  id: meta  
  uses: docker/metadata-action@v5  
  with:  
    images: ${{ vars.DOCKERHUB_USERNAME }}/example2  
    flavor: latest=false  
    tags: |  
      type=sha,priority=100,prefix=sha-,format=short  
      type=semver,priority=200,pattern={{version}}
```

Domyślnie, obraz jest tag-owany zgodnie z przyjętym schematem + generowany jest tag `latest`. Wartość `latest=false` wyłącza generowanie dodatkowego tag-u `latest`

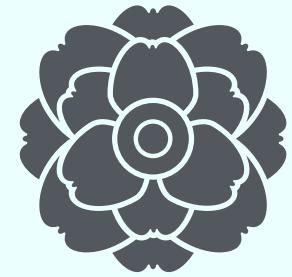
Metadata Action pozwala konfigurować tagowanie obrazów według trzech schematów:

- [Usage](#)
  - [Basic](#)
  - [Semver](#)
  - [Bake definition](#)

Event	Ref	Docker Tags
pull_request	refs/pull/2/merge	pr-2
push	refs/heads/master	master
push	refs/heads/releases/v1	releases-v1
push tag	refs/tags/v1.2.3	1.2.3, 1.2, latest
push tag	refs/tags/v2.0.8-beta.67	2.0.8-beta.67



Każdy schemat pozwala na wygenerowanie określonego tag-u w zależności od użytego trigger-a (od rodzaju event-u)



## Łańcuch CI - Tagowanie obrazów - cz. III

```
on:  
  workflow_dispatch:  
  push:  
    tags:  
      - 'v*'
```

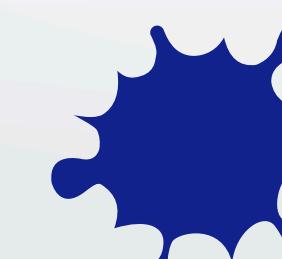
### Zdeklarowane zdarzenia (event-y)

Schemat sha (hash commit-a) zareaguje tak na event „dispatch” jak i „push”. W przypadku event-u „push” wyższy priorytet ma semver i on będzie tag-ował obraz.

Schemat semver „nie reaguje” na event dispatch, jedynie zareaguje na push tag-a

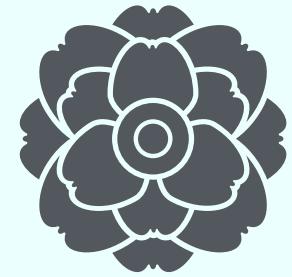
```
-  
  name: Docker metadata definitions  
  id: meta  
  uses: docker/metadata-action@v5  
  with:  
    images: ${{ vars.DOCKERHUB_USERNAME }}/example2  
    flavor: latest=false  
    tags: |  
      type=sha,priority=100,prefix=sha-,format=short  
      type=semver,priority=200,pattern={{version}}
```

Tagowanie zależy od rodzaju eventu a dodatkowo można poszczególnym schematom tag-owania przypisać priorytety. Im wyższy priorytet tym „ważniejszy” schemat.



Domyślna konfiguracja posiada przypisane wartości priorytetów, które można mienić jak wyżej.

<https://github.com/marketplace/actions/docker-metadata-action#priority-attribute>



# PAwChO – Laboratorium 9

## Łańcuch CI - Budowanie obrazów - cz. I

Marketplace / Actions / Build and push Docker images

 GitHub Action

**Build and push Docker images**

v5.1.0 Latest version

release v5.1.0 marketplace build-and-push-docker-images ci passing test passing coverage 57%

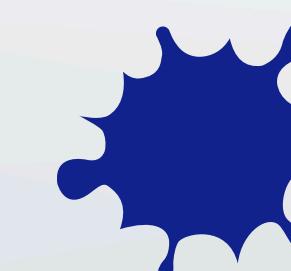
**About**

GitHub Action to build and push Docker images with [Buildx](#) with full support of the features provided by [Moby BuildKit](#) builder toolkit. This includes multi-platform build, secrets, remote cache, etc. and different builder deployment/namespacing options.

```
name: Build and push Docker image
uses: docker/build-push-action@v3
with:
  context: .
  file: ./Dockerfile
  platforms: linux/amd64,linux/arm64
  push: true
  cache-from: |
    type=registry,ref=${{ vars.DOCKERHUB_USERNAME }}/example2:cache
  cache-to: |
    type=registry,ref=${{ vars.DOCKERHUB_USERNAME }}/example2:cache
  tags: ${{ steps.meta.outputs.tags }}
```

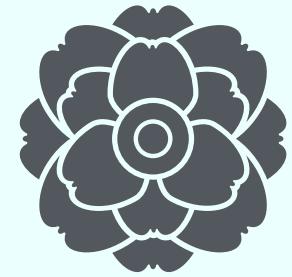
<https://github.com/marketplace/actions/build-and-push-docker-images>

Definicja docelowych architektur sprzętowych (analogicznie jak przy poleceniu docker buildx ...)



Definicja parametrów budowania obrazu (analogicznie jak przy poleceniu docker build ...)

W przypadku domyślnej nazwy pliku Dockerfile - wartość file: .... można pominąć



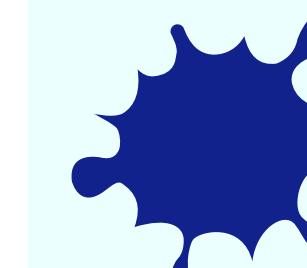
## Łańcuch CI - Budowanie obrazów - cz. II

```
name: Build and push Docker image
uses: docker/build-push-action@v3
with:
  context: .
  file: ./Dockerfile
  platforms: linux/amd64,linux/arm64
  push: true
  cache-from: |
    type=registry,ref=${{ vars.DOCKERHUB_USERNAME }}/example2:cache
  cache-to: |
    type=registry,ref=${{ vars.DOCKERHUB_USERNAME }}/example2:cache
  tags: ${{ steps.meta.outputs.tags }}
```



Pobranie z kroku „meta” wygenerowanych tag-ów i przypisanie ich do budowanego obrazu

Przesyłanie i pobieranie danych cache, które zgodnie z informacjami z poprzednich wykładów, pozwalają na przyśpieszenie operacji budowania kolejnych wersji obrazu danej aplikacji.



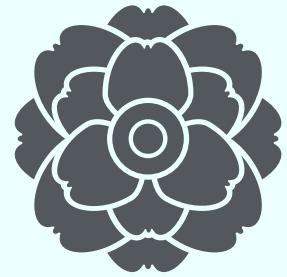
Proszę KONIECZNIE zapoznać się z przykładami z dokumentacji, szczególnie z typami cache oraz sposobami korzystania z każdego z nich.

<https://docs.docker.com/build/ci/github-actions/cache/>



Bardzo przydatna dokumentacja wykorzystania GitHub Actions w połączeniu z Docker

<https://docs.docker.com/build/ci/github-actions/>



# PAwChO – Laboratorium 9

## Github Actions + gh CLI - cz. I

```
~/Labs/lab9 🌟 main [✓]
```

```
> gh help actions
```

Welcome to GitHub Actions on the command line.

GitHub CLI integrates with GitHub Actions to help you manage runs and workflows.

### Interacting with workflow runs

```
gh run list:      List recent workflow runs
gh run view:     View details for a workflow run or one of its jobs
gh run watch:    Watch a workflow run while it executes
gh run rerun:    Rerun a failed workflow run
gh run download: Download artifacts generated by runs
```

To see more help, run `gh help run <subcommand>`

### Interacting with workflow files

```
gh workflow list:  List workflow files in your repository
gh workflow view:  View details for a workflow file
gh workflow enable: Enable a workflow file
gh workflow disable: Disable a workflow file
gh workflow run:   Trigger a workflow_dispatch run for a workflow file
```

To see more help, run `gh help workflow <subcommand>`

### Interacting with the GitHub Actions cache

```
gh cache list:    List all the caches saved in GitHub Actions for a repository
gh cache delete:  Delete one or all saved caches in GitHub Actions for a repository
```

To see more help, run `gh help cache <subcommand>`

GitHub CLI pozwala na sterowanie działaniem GitHub Actions za pomocą trzech zestawów poleceń

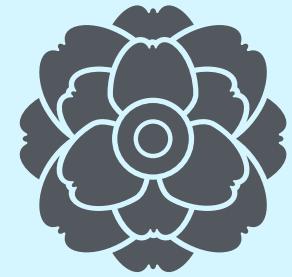
### Przykład działania:

```
~/Labs/lab9 🌟 main [✓]
> gh workflow list
```

NAME	STATE	ID
GHAction example	active	97769502
(END)		



Do danego łańcucha CI (workflow) można odwoływać się poprzez jego numer



## Github Actions + gh CLI - cz. II

```
apple ~ /Labs/lab9 🏙 main [✓]
> gh help workflow
List, view, and run workflows in GitHub Actions.

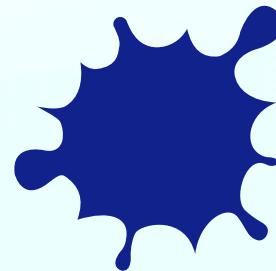
USAGE
  gh workflow <command> [flags]

AVAILABLE COMMANDS
  disable:      Disable a workflow
  enable:       Enable a workflow
  list:         List workflows
  run:          Run a workflow by creating a workflow_dispatch event
  view:         View the summary of a workflow
```

Polecenie `gh workflow run` uruchamia działanie łańcucha CI WYŁĄCZNIE jeśli zdeklarowany jest wyzwalacz (event) w postaci `workflow_dispatch`

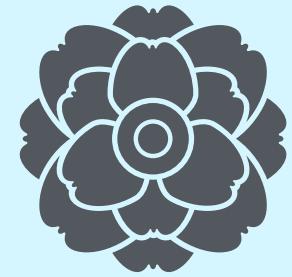
W przykładowym łańcuchu CI dla Github Actions zdeklarowane są dwa wyzwalacze:

- przesłanie nowego tag-u do repo git (gałąź main): uruchamia tagowanie obrazu wg. Semver
- Ręczne uruchomienie łańcucha CI: uruchamia tagowania bazujące na sha ostatniego commit-a (na gałęzi main)



Pełna dokumentacja metod wyzwalania workflows:

<https://docs.github.com/en/actions/using-workflows/triggering-a-workflow>



## Github Actions - użycie workflow\_dispatch - cz. I

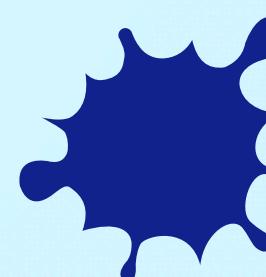
1

Uruchomienie  
łańcucha CI



```
🍎 ~/Labs/lab9 🌐 main [✓]
> gh workflow run
? Select a workflow GHAction example (gha_example.yml)
✓ Created workflow_dispatch event for gha_example.yml at main
```

To see runs for this workflow, try: `gh run list --workflow=gha_example.yml`



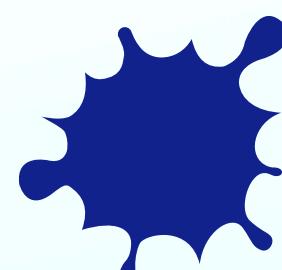
Polecenie w ramach `gh` można używać iteraktywnie (jak w przykładzie) lub poprzez deklarowanie wszystkich argumentów w ramach danego polecenia (proszę sprawdzić pomoc dla wybranego polecenia)

2

Sprawdzenie statusu  
wykonania łańcucha CI



```
🍎 ~/Labs/lab9 🌐 main [✓]
> gh run view
? Select a workflow run ✓ GHAction example, GHAction example (main)
```



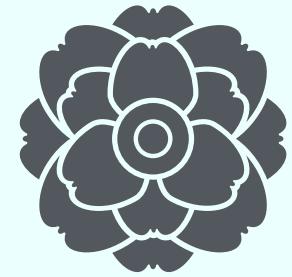
W trakcie realizacji  
danego workflow można  
obserwować za pomocą  
polecenia `gh run watch`

```
✓ main GHAction example · 9054534590
Triggered via workflow_dispatch about 6 minutes ago

JOBS
✓ Build, tag and push Docker image to DockerHub in 28s (ID 24874509225)

For more information about the job, try: gh run view --job=24874509225
View this run on GitHub: https://github.com/SenatorP51/lab9/actions/runs/9054534590

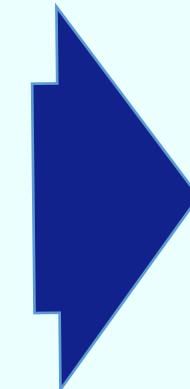
(END)
```



## Github Actions - użycie workflow\_dispatch - cz. II

3

Szczegóły realizacji danego łańcucha CI



```
🍎 ~/Labs/lab9 🏃 main [✓]
> gh run view --job=24874509225
```

✓ main GHAction example · 9054534590

Triggered via workflow\_dispatch about 2 minutes ago

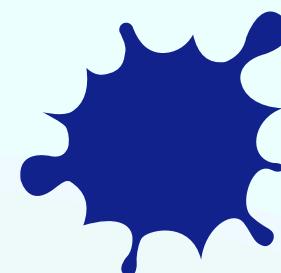
✓ Build, tag and push Docker image to DockerHub in 28s (ID 24874509225)

- ✓ Set up job
- ✓ Check out the source\_repo
- ✓ Docker metadata definitions
- ✓ QEMU set-up
- ✓ Buildx set-up
- ✓ Login to DockerHub
- ✓ Build and push Docker image
- ✓ Post Build and push Docker image
- ✓ Post Login to DockerHub
- ✓ Post Buildx set-up
- ✓ Post Check out the source\_repo
- ✓ Complete job

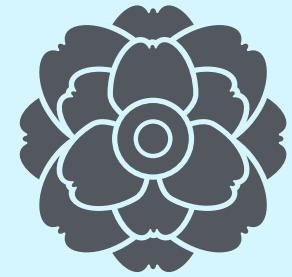
To see the full job log, try: gh run view --log --job=24874509225

View this run on GitHub: <https://github.com/SenatorP51/lab9/actions/runs/9054534590>

(END)



Szczególnie w przypadku wystąpienia błędów  
WARTO pamiętać o możliwości wyświetlenia bardzo szczegółowych logów

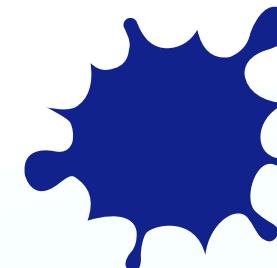


# PAwChO – Laboratorium 9

## Github Actions - użycie workflow\_dispatch - cz. III

Sprawdzenie poprawności wykonanych działań:

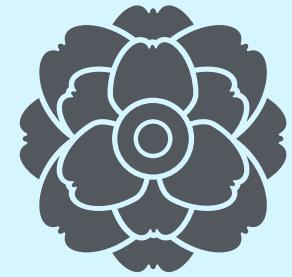
The screenshot shows the GitHub Actions interface. On the left, there's a sidebar with options like 'Actions', 'New workflow', 'All workflows' (which is selected), 'GHAction example', 'Management', 'Caches', 'Attestations', and 'Runners'. The main area is titled 'All workflows' with the subtitle 'Showing runs from all workflows'. It shows '1 workflow run' for 'GHAction example'. A red box highlights the green checkmark icon next to the workflow name. Below it, it says 'GHAction example #1: Manually run by SenatorP51'. To the right, the word 'Github' is written in large blue letters.



Polecenia z grupy `gh cache` ...  
pozwalają na operacje na danych cache  
TYLKO jeśli te dane są przechowywane  
na repozytorium `ghcr.io`

The screenshot shows the Dockerhub repository page for 'spg51/example'. At the top, it says 'spg51/example' (with a star icon), 'Updated 24 minutes ago', and 'GHActions example' with a pencil icon. Below that, it says 'This repository does not have a category' with a pencil icon and 'INCOMPLETE' with an info icon. To the right, it says 'Analyzed by docker scout'. The page has sections for 'Tags' and 'Images'. The 'Tags' section says 'This repository contains 2 tag(s)'. The 'Images' section has a table with columns: Tag, OS, Type, Vulnerabilities, Pulled, and Pushed. Two rows are shown: one for tag 'cache' (OS: ---, Type: Image, Pulled: ---, Pushed: 24 minutes ago) and one for tag 'sha-1de15d5' (OS: Alpine, Type: Image, Vulnerabilities: None found, Pulled: ---, Pushed: 24 minutes ago). Red boxes highlight the 'cache' row and the 'sha-1de15d5' row. To the right, the word 'Dockerhub' is written in large blue letters.

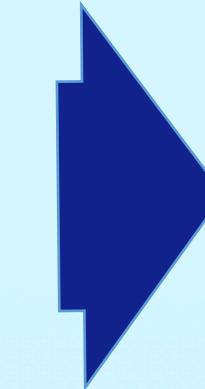
Tag	OS	Type	Vulnerabilities	Pulled	Pushed
cache	---	Image		---	24 minutes ago
sha-1de15d5	Alpine	Image	None found	---	24 minutes ago



## Github Actions - użycie git tags - cz. I

1

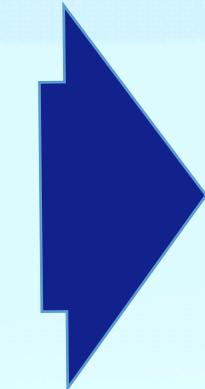
Utworzenie nowego tag-a na gałęzi main na lokalnym repozytorium git



```
apple ~ ~/Labs/lab9 ⚡ main [✓]
> git tag -a "v1.0.1" -m "test version v1.0.1"
```

2

Przesłanie tag-a na repo na Github (synchronizacja)



```
apple ~ ~/Labs/lab9 ⚡ main [✓]
> git push origin tag v1.0.1
Wymienianie obiektów: 1, gotowe.
Zliczanie obiektów: 100% (1/1), gotowe.
Zapisywanie obiektów: 100% (1/1), 183 bajty | 183.00 KiB/s, gotowe.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:SenatorP51/lab9.git
* [new tag]           v1.0.1 -> v1.0.1
```

Refreshing run status every 3 seconds. Press Ctrl+C to quit.

```
* v1.0.1 GHAction example · 9059877631
Triggered via push less than a minute ago
```

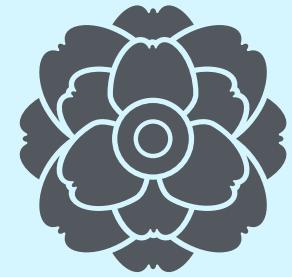
### JOB

```
* Build, tag and push Docker image to DockerHub (ID 24888392878)
  ✓ Set up job
  ✓ Check out the source_repo
  ✓ Docker metadata definitions
  ✓ QEMU set-up
  ✓ Buildx set-up
  ✓ Login to DockerHub
  ✓ Build and push Docker image
  ✓ Post Buildx set-up
  ✓ Post Check out the source_repo
```

3

Obserwacja „na żywo” działań zdefiniowanych w ramach łańcucha CI

```
apple ~ ~/Labs/lab9 ⚡ main [✓]
> gh run watch
? Select a workflow run * Initial config, GHAction example (v1.0.1)
✓ v1.0.1 GHAction example · 9059877631
```



# PAwChO – Laboratorium 9

## Github Actions - użycie git tags - cz. II

Sprawdzenie poprawności wykonanych działań:

**Dockerhub**

spg51/example

Updated 12 minutes ago

GHActions example

This repository does not have a category INCOMPLETE

**Tags**

This repository contains 3 tag(s).

Tag	OS	Type	Vulnerabilities	Pulled	Pushed
cache	---	Image		---	12 minutes ago
sha-1de15d5		Image	None found	13 minutes ago	13 minutes ago
1.0.1		Image	None found	13 minutes ago	13 minutes ago

Analyzed by docker scout

**gh CLI**

```
~/Labs/lab9 🌐 main [✓]
> gh run view
? Select a workflow run ✓ Initial config, GHAction example (v1.0.1)

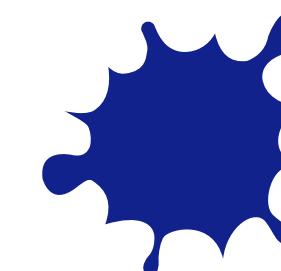
✓ v1.0.1 GHAction example · 9059877631
Triggered via push about 9 minutes ago
```

JOB  
✓ Build, tag and push Docker image to DockerHub in 20s (ID 24888392878)

For more information about the job, try: gh run view --job=24888392878  
View this run on GitHub: [\(END\)](https://github.com/SenatorP51/lab9/actions/runs/9059877631)

```
~/Labs/lab9 🌐 main [✓]
> gh run view
? Select a workflow run ✓ Initial config, GHAction example (v1.0.1)

✓ v1.0.1 GHAction example · 9059877631
Triggered via push about 9 minutes ago
```



Poprzez sterowanie doboru wyzwalaczy można tworzyć obrazy o tag-owaniu wymaganym przez ewentualne, kolejne etapy działań w ramach łańcucha (np, testy czy CD)