Z A D A N I E N R 1

Podstawą do wykonania zadań w zadaniu nr 1 jest aplikacja **FibCalc**.

CZĘŚĆ OBOWIĄZKOWA

1. Jako pierwszy krok należy napisać aplikację, która:

A: przyjmuje od użytkownika wartość całkowitą, która jest wykorzystywana jako numer elementu ciągu Fibonnaciego (https://en.wikipedia.org/wiki/Fibonacci_number),

B: wylicza w oparciu o dowolny, wybrany algorytm, wartość ciągu dla podanego elementu,

C: wyświetla otrzymany wynik oraz następujące informacje dodatkowe:

1) nazwę programu,
2)imię i nazwisko autora,
3) numer grupy dziekańskiej.

Kod aplikacji ma znajdować się na GitHub, w dowolnym repo publicznym studenta. Utworzenie tego repo należy wykonać za pomocą odpowiednich poleceń: *git* oraz *gh*. Należy podać te polecenia.

UWAGA1: Dobór algorytmu jak i UI/GUI programu należy do studenta.
UWAGA2: Wybór języka i środowiska, w którym zostanie przygotowana aplikacja również jest wyłączną decyzją studenta.

2. Na bazie opracowanego kodu należy:

A: przygotować plik *Dockerfile*, który umożliwi zbudowanie obrazu zawierającego kod opracowanego programu i wszystkie niezbędne zależności oraz wpisy definiujące sposób uruchomienia kontenera na bazie tego obrazu,

B: zbudować obraz wykorzystując dowolną metodę omawianą na laboratorium oraz podać wykorzystane polecenie,

C: podać wykorzystane polecenie i zaprezentować efekt działania skonteneryzowanej aplikacji (potwierdzić poprawność działania kodu).

3. Po stwierdzeniu, że aplikacja działa poprawnie oraz, że procedura budowania obrazy kończy się sukcesem, należy:

A: przygotować plik definiujący ciąg działać (job) w ramach GitHub Action. Plik ten powinien nazywać się *fib.yml*. Jego zawartość powinna gwarantować:

1) plik *Dockerfile* przygotowany w punkcie 2A ma być podstawą do zbudowania obrazy na węźle (runner) z systemem operacyjnym Ubuntu 20.04,
2) proces budowy obrazu zrealizowany ma być w oparciu o silnik buildkit,
3) obrazy mają umożliwić uruchomienie aplikacji FibCalc na komputerach z procesorami 64-bitowymi Intel oraz Apple M1,
4) obrazy mają być zgodne ze specyfikacją OCI.

B: W ramach działania workflow GitHub Action należy zbudowane obrazy przesłać do swojego repozytorium PUBLICZNEGO na GitHub Packages (repo: ghcr.io).

C: W trakcie budowania obrazów należy wykorzystywać informacje cache w trybie registry a docelowym repozytorium ma być publiczne repo studenta na DockerHub.
D: W pliku *fib.yml* należy zdeklarować i wykorzystać zasadę nazywania budowanych obrazów według metody.

UWAGA: Podczas realizacji punktu 3D należy wykorzystać informację zawarte pod poniższymi adresami internetowymi:
https://semver.org/lang/pl/
https://github.com/docker/metadata-action#semver

4. Za pomocą narzędzia *gh* należy:
A: Sprawdzić obecność pliku *fib.yml* jako opisu workflow w GitHub Action.
B: Uruchomić GitHub Action i potwierdzić poprawność działania opracowanego rozwiązania.
C: Pobrać wybrany obraz (obraz na architekturę wykorzystywaną na swoim komputerze) i uruchomi kontener z opracowaną aplikacją. Potwierdzi poprawność działania aplikacji.

Sprawozdanie z części podstawowej może mieć dwie formy (do wyboru):

1. Plik z linkiem do utworzonego i wykorzystywanego przy wykonywaniu zadania repozytorium na GitHub. Na tym repozytorium powinny być utworzone jeden lub więcej (do poszczególnych punktów) pliki **.md*.
2. Klasyczny plik dot/doc/pdf z opisem przygotowanego rozwiązania oraz linkiem do do utworzonego i wykorzystywanego przy wykonywaniu zadania repozytorium na GitHub.

W każdym wypadku opis rozwiązania należy wgrać do przygotowanego katalogu na moodle.

Zawartość sprawozdania:

Ad. p1. Należy podać link do repozytorium publicznego na GitHub, krótkie omówienie algorytmu wykorzystanego w programie FibCalc oraz listę poleceń tworzących środowisko pracy na GitHub z efektem ich działania.

Ad. p2. Należy podać polecenia do zbudowania obrazu, jego uruchomienia wraz z efektami ich działania oraz dowód na poprawność działania aplikacji z założeniami z punktu 1.

Ad. p3. Wystarczy sam plik fib.yml na repozytorium GitHub oraz krótki opis przyjętej realizacji nazewnictwa obrazów zgodnie z metodą *semver* oraz zasady wykorzystania repo ghcr.io w pliku *fib.yml*.

Ad. p4. Należy podać wszystkie wykorzystane polecenia wraz z efektami ich działania. Opracowany workflow należy uruchomić minimum dwa razy i na tej podstawie przedstawić dowód na poprawne działanie cache oraz automatycznego generowania nazw obrazów zgodnie z założeniami w p. 3. Dodatkowo należy podać link do

repozytorium publicznego na ghcr.io oraz do repozytorium publicznego na docker.io, które wykorzystywane były w pliku ib.yml

Maksymalna ilość punktów z części obowiązkowej – 100%
Oceny cząstkowe:
p. 1 – 20%
p.2 – 10%
p.3 – 40%
p.4 – 30%

Dodatkowo, osoba, które w ramach danej grupy zbuduje najmniejszy obraz z poprawnie działającą aplikacją (pod uwagę będzie brany obraz dla architektury 64-bitowej Intel) otrzyma **ekstra 50% punktów.**

CZĘŚĆ NIEOBOWIĄZKOWA

Część ta zawiera dwa tematy. Można zrealizować jeden (wybrany) temat lub oba. W tym drugim przypadku punkty sumują się.

Zadanie nieobowiązkowe 1

Jeżeli program napisany na potrzeby tego zadania wymaga kompilacji i/lub powinno się w pliku Dockerfile uwzględnić specyfikę budowy obrazu dla różnych platform sprzętowych to:

- należy przygotować zmodyfikowany plik Dockerfile o nazwie *Dockerfile_dod1* zawierający odpowiednie deklaracje, które powinny być wskazane poprzez odpowiednie komentarze w treści pliku.

- użyć tego pliku w przygotowanym workflow (proszę nazwać go *fib_dod1.yml*) i potwierdzić poprawność tak działania zmodyfikowanego GitHub Actions jak i możliwości uruchomienia zbudowanego obrazu i uruchomienia aplikacji FibCalc (analogicznie jak w punkcie 4 zadania podstawowego).

UWAGA: W przygotowaniu tego rozwiązania pomocny będzie link podany na jednym z poprzednich laboratoriów, który podany jest również poniżej:
https://www.docker.com/blog/faster-multi-platform-builds-dockerfile-cross-compilation-guide/

Maksymalna ilość punktów z nieobowiązkowego zadania nr 1 – 100%

Zadanie nieobowiązkowe 2

W ramach tego zadania należy utworzyć własne repozytorium na bazie wykorzystywanego już oficjalnego już obrazy *registry.v2*. Repozytorium to należy tak skonfigurować aby możliwe było jego wykorzystanie do przechowywania danych cache (zamiast repo na DockerHub jak to jest wskazane w zadaniu podstawowym)

- należy przedstawić procedurę konfiguracji repo wykonanej w trakcie realizacji tego zadania,

- należy zmodyfikować przygotowany w zadaniu podstawowym workflow (proszę nazwać go *fib_dod2.yml*) i potwierdzić poprawność tak działania zmodyfikowanego GitHub Actions jak i poprawności operacji na danych cache (analogicznie jak w punkcie 4 zadania podstawowego).

UWAGA: W przygotowaniu tego rozwiązania pomocne będą linki podane na jednym z poprzednich laboratoriów, które podane są również poniżej:
https://hub.docker.com/_/registry
https://docs.docker.com/registry/deploying/

Maksymalna ilość punktów z nieobowiązkowego zadania nr 1 – 100%

UWAGI KOŃCOWE:

1. Zadanie należy wykonać SAMODZIELNIE. W przypadku kopii (tak typu Ctr C – Ctr V, jak i modyfikacji nie istotnych merytorycznie) ocena z zadania zostanie PODZIELONA przez liczbę „współtwórców".
2. Oceniający zastrzega sobie prawo do podwyższenia oceny jeśli dane rozwiązanie chwyci go za serce albo umysł.