***Akademia Marynarki Wojennej***
*Katedra Informatyki*
*Wydział Mechaniczno-Elektryczny*


**Przedmiot:** Aplikacje w architekturze klient-serwer

# Skrypt Laboratoryjny

Mgr inż. Maciej Szulc, md.szulc@amw.gdynia.pl
Mgr. inż. Bartosz Wawrzyniak, b.wawrzyniak@amw.gdynia.pl
Gdynia, 2022

## *Laboratorium nr.* **15**

| | |
|---|---|
| ***Temat zajęć:*** | *Konteneryzujemy aplikację - mini projekt* |
| ***Prowadzący*** | *Maciej Szulc, md.szulc@amw.gdynia.pl*<br>*Bartosz Wawrzyniak, b.wawrzyniak@amw.gdynia.pl* |
| ***Forma zajęć*** | *Laboratorium praktyczne* |
| ***Forma zaliczenia*** | *Poprawne wykonanie ćwiczeń zgodnie z wymogami przedstawionymi w niniejszym skrypcie.* |
| ***Uwagi dodatkowe*** | |

# FORMAT RAPORTU Z LABORATORIUM.

Na początku raportu należy zamieścić nagłówek o formacie:

| | | | |
|---|---|---|---|
| **Laboratorium nr.:** | | **Data laboratorium:** | |
| **Temat:** | | | |
| **Autor raportu:** | | | |
| **Grupa:** | | | |
| **Grupa lab.** | | **Data raportu:** | |
| **Subiektywna ocena trudności laboratorium (nie wpływa na ocenę pracy!) [1-łatwe, 10-trudne]** | | | |

Aby "zaliczyć" laboratorium musisz dostarczyć raport przygotowany w dowolnym edytorze zawierający:

1. Nagłówek wg. specyfikacji powyżej
2. Odpowiedzi na wszystkie pytania laboratoryjne, wraz numeracją każdego pytania (w formacie LITERA.CYFRA, np. A.1).
3. **Opcjonalną** sekcję, w której opiszesz napotkane problemy i sposób ich rozwiązania

Raport ten należy wgrać na Platformę Edukacyjną AMW (Przedmiot: "Aplikacje w architekturze Klient Serwer - Laboratorium", element: "Tu wgraj raport z Laboratorium nr. X").

**Proszę NIE WYSYŁAĆ raportów mailem.**

**Akceptowane formaty raportów: PDF**

# Mini projekt

W ramach prac będziesz odpowiedzialny za:

- napisanie kodu aplikacji
- skonteneryzowanie aplikacji
- wgranie jej obrazu do repozytorium prywatnego na klastrze Kubernetes
- przygotowanie plików yaml niezbędnych do jej uruchomienia
- wdrożenie i uruchomienie aplikacji
- przetestowanie jej działania wg. przygotowanego scenariusza testów

Plan zajęć związanych z mini-projektem jest następujący:

- **21.01.2022**: pisanie kodu aplikacji, konteneryzacja aplikacji i wstępne uruchomienie bazy danych
- **26.01.2022**: ew. dokończenie procesu konteneryzacji, finalne uruchomienie bazy danych, przygotowanie obiektów Kubernetes: Deployment, ConfigMap, Secret, PVC i inne. Na te zajęcia powinieneś mieć niemal końcową wersję aplikacji. Musisz w tym terminie przekazać również propozycję finalnej listy testów aplikacji.
- **28.01.2022**: końcowe uruchomienie, testy, poprawki, zaliczenie - na te zajęcia musisz mieć finalną wersję aplikacji (jeśli zależy Ci na wyższej ocenie).

  **31.01.2022**: dodatkowy, ostateczny termin oddawania prac, które nie zostały zaliczone na zajęciach 28.01.2022.

  **Oddanie aplikacji w terminie późniejszym będzie skutkowało zmniejszeniem oceny**.

Przy wyliczaniu oceny końcowej za laboratorium przedmiotowe ocena za mini-projekt będzie wliczana z wagą równą 5 "tradycyjnym" zajęciom laboratoryjnym.

**UWAGA: Dzisiejszy lab podlega zaliczeniu (lub nie), nie jest oceniany według skali procentowej!!!**

# Budowa obrazu aplikacji

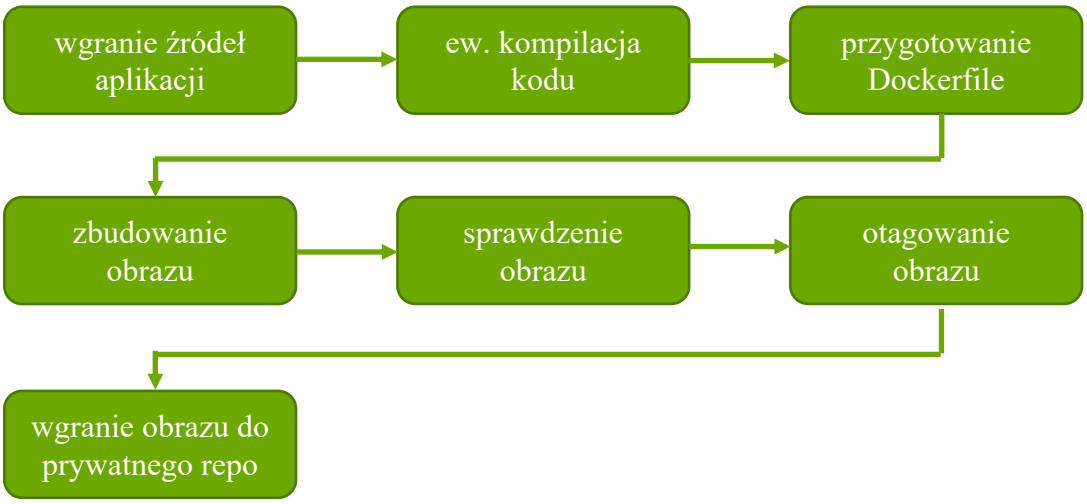
W ramach dzisiejszego laboratorium odpowiedzialny będziesz za zbudowanie obrazów poszczególnych modułów Twojej aplikacji. Instrukcje jakie otrzymasz będą bardziej wskazówkami, niż gotowym przepisem na konteneryzację Twojej aplikacji - nasz mini projekt jest bowiem pomyślany jako forma podsumowania i weryfikacji nabytych na zajęciach umiejętności.

Po utworzeniu obrazów będziesz musiał je przetestować, otagować a następnie wgrać do repozytorium obrazów na klastrze Kubernetes.

Pod koniec zajęć powinieneś również zająć się kwestią bazy danych - o ile jest ona wymagana dla Twojej aplikacji.

W razie wątpliwości sięgnij do skryptów z poprzednich laboratoriów.

Przebieg dzisiejszych prac będzie następujący:



# Logowanie do klastra

1. (PC) Zaloguj się przez SSH na serwer Linux

# Katalog na serwerze Linux i kod źródłowy

2. (LINUX) Utwórz katalog "projekt" i wejdź do niego

```
mkdir projekt
cd projekt
```

3. (LINUX) Do utworzonego katalogu wgraj kod źródłowy Twojej aplikacji

# Kompilacja kodu, przygotowanie Dockerfile i utworzenie obrazu

Zanim przygotujesz odpowiedni plik Dockerfile i wykonasz całą operację budowy obrazu - być może będziesz musiał (w zależności od rodzaju środowiska w jakim stworzyłeś swoją aplikację) najpierw ją skompilować. Do obrazu musi trafić aplikacja gotowa do uruchomienia.

Ze względu na mnogość możliwych realizacji Twojej aplikacji nie mam możliwości podania w tym punkcie zbyt szczegółowych instrukcji jak wykonać te operacje, ale wiedza jaką nabyłeś podczas dotychczasowych zajęć i instrukcje dostępne w sieci powinny pomóc wykonać zadanie.

Tworząc dockerfile nie zapomnij o poprawnych argumentach uruchomienia (CMD/ENTRYPOINT) oraz udostępnieniu portów aplikacji (EXPOSE)

### dla aplikacji Nodejs

Posłuż się np. instrukcją z Laboratorium nr. 2 - " **Konteryzacja aplikacji Node.JS"**

### dla aplikacji Spring

Posłuż się np. instrukcją dostępną pod adresem: https://spring.io/guides/gs/spring-boot-docker/.

Na naszym serwerze zainstalowany jest zarówno Gradle, jak i Maven.

Jeśli w procesie budowy aplikacji uwzględnione są testy wymagające zewnętrznych zasobów (np. podłączenia do bazy danych) to na czas budowy aplikacji należy je wyłączyć - w przeciwnym wypadku aplikacja zapewne się nie zbuduje.

Uwaga: jako obraz startowy (linia FROM w Dockerfile) warto użyć nowszy niż ten zawarty w powyższej instrukcji. Na ogół dobrym wyborem będzie:

```
FROM openjdk:latest
```

### dla aplikacji Python

Posłuż się np. instrukcją dostępną pod adresem:

**https://www.docker.com/blog/containerized-python-development-part-1**

***Q-1. W raporcie zamieść log z budowy obrazu aplikacji***

# Testowanie obrazu aplikacji

4. (LINUX) Sprawdź, czy obrazy zostały wybudowane i są dostępne:

```
docker images
```

5. (LINUX) Każdy ze zbudowanych obrazów przetestuj w środowisku lokalnym poprzez uruchomienie kontenera poleceniem "docker run". Wykorzystaj opcje:

   - -p host_port:Container_port - uodstępniającą port aplikacji
   - -d - uruchamiającą aplikację w tle (jeśli chcesz uruchomić w tle)
   - --name - nadającą nazwę kontenerowi

   np.:

```
docker run -p <port_host>:<port_kontener> --name <nazwa> -d <nazwa_obrazu>:<wersja_obrazu>
```

6. (LINUX) Sprawdź logi Twojej aplikacji

```
docker logs <nazwa|numer kontenera>
```

7. (LINUX) Przetestuj dostępność skonteneryzowanej aplikacji poleceniami curl/wget/lynx, np.:

```
curl  http://localhost:<port hosta>/
```

**8.** (LINUX) Jeśli aplikacja działa poprawnie: zatrzymaj (stop) i usuń (rm) kontener aplikacji Jeśli natomiast występują błędy - popraw je, zbuduj nową wersję obrazu i ją ponownie przetestuj. Możesz również wykorzystać polecenie "docker exec" i "wejść" do środka kontenera aby wykonać czynności diagnostyczne.

# Tagowanie obrazów

9. (LINUX) Sprawdźmy, jakie mamy **lokalnie** obrazy poleceniem

```
docker images
```

przykładowa odpowiedź zawierać będzie nazwy i wersje obrazów, np. jak poniżej

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mywww               1.1                 d7c4df81f4bb        3 days ago          943MB
mywww               1.0                 81acbb143b5e        3 days ago          943MB
<none>              <none>              4579e031f2f4        4 days ago          943MB
node                latest              ca36fba5ad66        12 days ago         941MB
```

10. (LINUX) Aby wysłać obrazy do zdalnego repozytorium na klastrze Kubernetes trzeba je oznaczyć (tag) jego identyfikatorem (adresem).

polecenie służące do tagowania obrazów ma następującą składnię:

```
docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]
```

gdzie:

- **SOURCE_IMAGE** - to nazwa lokalnego obrazu (np. mywww)
- **TAG** - to wersja obrazu (np. 1.0)
- **TARGET_IMAGE** - to adres repozytorium wraz z katalogiem, do którego obraz ma być wgrany.
  W naszym przypadku będzie to
  **default-route-openshift-image-registry.apps.ocp.lab.cloudpak.site/<nazwa Twojego namespace>/<nazwa obrazu>:<TAG>**

Otagujmy zatem zbudowane uprzednio obrazy poniższym poleceniem
***uwaga:*** *zamień labprojXX na Twój identyfikator. Poniższa linia może być w niniejszym skrypcie zawinięta - ale Ty musisz ją wpisać w jednej, ciągłej linii!!!*

```
docker tag <obraz>:<wersja> default-route-openshift-image-
registry.apps.ocp.lab.cloudpak.site/labprojXX/<obraz>:<wersja>
```

11. (LINUX) Sprawdźmy poprawność tagowania poleceniem

```
docker images
```

przykładowa odpowiedź:

```
REPOSITORY                                                                  TAG
IMAGE ID            CREATED             SIZE
mywww                                                                       1.1
d7c4df81f4bb        3 days ago          943MB
default-route-openshift-image-registry.apps.ocp.lab.cloudpak.site/labproj25/mywww   1.0
81acbb143b5e        3 days ago          943MB
mywww                                                                       1.0
81acbb143b5e        3 days ago          943MB
<none>                                                                      <none>
4579e031f2f4        4 days ago          943MB
node                                                                        latest
ca36fba5ad66        12 days ago         941MB
```

10

Jak widać obraz został poprawnie otagowany - na liście są obrazy oznaczone nazwą naszego prywatnego repozytorium.

## Logowanie do klastra

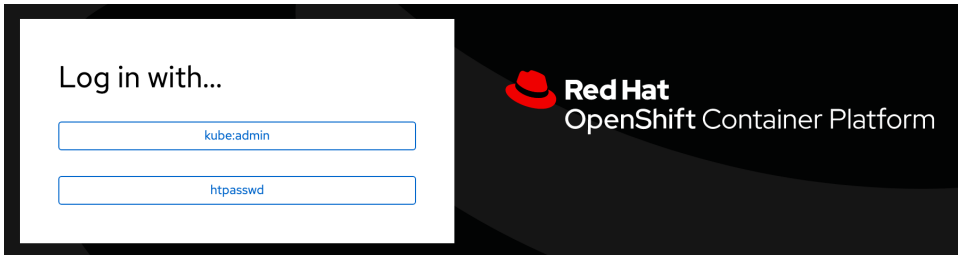
Zalogujemy się zarówno do interfejsu GUI (na Twoim PC), jak i do interfejsu CLI (na serwerze Linux)

### *GUI*

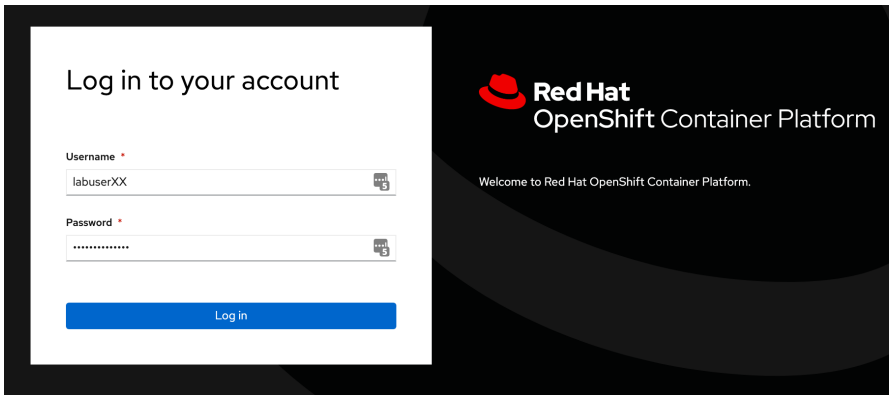
12. (PC) W dowolnej przeglądarce otwórz URL

```
https://console-openshift-console.apps.ocp.lab.cloudpak.site/
```

13. (PC) Wybierz pozycję "Log in with… htpasswd"




14. (PC) Zaloguj się wykorzystując otrzymane dane logowania (labuserXX) do klastra (uwaga: to nie są loginy do serwera Linux!)




### *CLI*

15. (LINUX) Do logowania się na klaster wykorzystamy polecenie "oc", które jednocześnie skonfiguruje nam narzędzie "kubectl".
*OC to interfejs CLI oferowany przez OpenShift. Kubectl to interfejs CLI z projektu Kubernetes.*

Pełna komenda:

```
oc login -u labuserXX -p <hasło> https://api.ocp.lab.cloudpak.site:6443
```

Zaloguj się na klaster używając swojego ID i hasła.

*jeśli zobaczysz zapytanie:*

```
The server uses a certificate signed by an unknown authority.
You can bypass the certificate check, but any data you send to the server could be intercepted by
others.
Use insecure connections? (y/n):
```

*potwierdź poprzez wpisanie y i zatwierdzenie enterem.*

16. (LINUX) Po zalogowaniu zmień aktywny projekt (namespace) na swój własny - labprojXX poleceniem:

```
kubectl config set-context --current --namespace labprojXX
```

17. **alternatywnie do powyższego** możesz użyć polecenia "oc project" (*uwaga: poniższe polecenie zadziała wyłącznie na klastrach OpenShift - więc jeśli w przyszłości będziesz chciał korzystać z klastra "vanilla" Kubernetes to polecenie nie zadziała*)

```
oc project labprojXX
```

# Wysyłanie obrazów do repozytorium

18. (LINUX) Aby polecenie "docker" miało możliwość wysyłania obrazów do repozytorium prywatnego musimy się najpierw zalogować poprzez "docker login".
    Wpisz i wykonaj polecenie:

```
docker login -u $(oc whoami) -p $(oc whoami -t) https://default-route-openshift-image-registry.apps.ocp.lab.cloudpak.site/
```

    znaczenie poszczególnych elementów komendy jest następujące:

    - **-u $(oc whoami)** - nazwa użytkownika. W naszym przypadku pobieramy ją automatycznie dzięki wykorzystaniu komendy "oc whoami", ale oczywiście można to zrobić również manualnie. Aby zadziałało poprawnie musisz być wcześniej zalogowany do klastra poleceniem "oc" - co wykonaliśmy chwilę temu.
    - **-p $(oc whoami -t)** - hasło. W naszym przyadku w formie tokena zwracanego przez "oc whoami -t"
    - https://default-route-openshift-image-registry.apps.ocp.lab.cloudpak.site/ - URL naszego repozytorium prywatnego

    Poprawne logowanie powinno zakończyć się komunikatem:

```
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /home/student/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

19. (LINUX) Czas na przesłanie obrazu do prywatnego repozytorium. Obrazy wysyłamy poleceniem "docker push" o następującej składni:

```
docker push [OPTIONS] NAME[:TAG]
```

    gdzie:

    - NAME - to nazwa obrazu (po otagowaniu)
    - TAG - wersja obrazu

    Wpisz zatem polecenie:

    ***uwaga:*** *zamień labprojXX na Twój identyfikator. Poniższa linia może być w niniejszym skrypcie zawinięta - ale Ty musisz ją wpisać w jednej, ciągłej linii!!!*

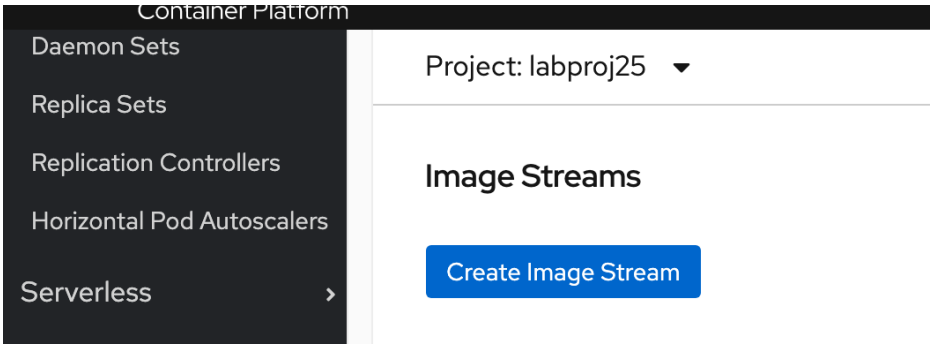
```
docker push default-route-openshift-image-registry.apps.ocp.lab.cloudpak.site/labprojXX/<obraz>:<wersja>
```

    i poczekaj na przesłanie.

    W ten sposób prześlij wszystkie Twoje obrazy

# Weryfikacja dostępności obrazów

20. (PC) Powróć do przeglądarki WWW zalogowanej do klastra. Przejdź do menu "**Builds**"->"**Image Streams**".
    Uwaga: Sprawdź, czy jesteś w swoim namespace (projekcie) labprojXX, w razie konieczności zmień projekt.




21. (PC) Sprawdź, czy na liście dostępnych obrazów są dostępne utworzone przez Ciebie obrazy

***Q-2. W raporcie zamieść zrzut ekranu z obrazami***

***Q-3 W raporcie zamieść adres repozytorium z kodem Twojej aplikacji***

# Baza danych

22. Jeśli Twoja aplikacja wymaga bazy danych - zajmij się przygotowaniem jej instalacji. Poszukaj w Internecie informacji o sposobie instalacji, w razie konieczności porozmawiaj o tym z wykładowcą.

Pamiętaj o przygotowaniu PVC w oparciu o dostępną storageclass "managed-nfs-storage"

Na naszym klastrze dostępne są bezpośrednio następujące technologie bazodanowe:

- MongoDB
- MySQL
- Postgres
- MariaDB

Instancje tych baz mogą zostać Tobie udostępnione przez prowadzącego zajęcia.

Alternatywnie możesz bazę uruchomić samodzielnie w swoim namespace.
Jeśli potrzebujesz obrazu którejś z baz danych, pamiętaj o ściągnięciu jej z dockerhub. (Nie zapomnij zalogować się poleceniem docker login na swoje prywatne konto). Następnie otaguj pobrany obraz i wyślij do prywatnego repo na klastrze.

***Q-4. W raporcie zamieść informację, jaką bazę danych (technologia, wersja) będzie wykorzystywała Twoja aplikacja***