Projekt zaliczeniowy II

Wymagania dotyczące formuły:

- Commit na dedykowanej gałęzi w repozytorium MDO2022
- Sprawozdanie stosujące język Markdown (MD), z zagnieżdżonymi w tekście zrzutami ekranu z procesu
- Dostępne w repozytorium pliki `Dockerfile, Jenkinsfile`, kompozycje, logi z *builda* – wszystkie na które powołano się w treści sprawozdania
- Zastosowane polecenia wklejone, jako obiekt `kod`, do sprawozdania
- Pliki `Dockerfile, Jenkinsfile` mają umożliwiać ponowienie opisywanych w sprawozdaniu operacji podczas sprawdzania
- Jeżeli czynność nie jest odtworzeniem operacji podręcznikowej i wymaga podjęcia decyzji charakteryzującej dany krok, konieczne jest opisanie motywacji za decyzją

Zadanie:

- Znaleźć wśród repozytoriów GitHuba projekt zawierający mechanizm budowania (`Automake, Meson, npm`…) oraz testy jednostkowe.
    - Przykładem niech będzie `irssi`, ale należy użyć innego projektu.
    - Licencja projektu powinna czynić projekt możliwym do *forkowania* na własne potrzeby przy zasadzie udostępniania dzieł pokrewnych i wywiedzionych

- Wykazać w środowisku Docker, że program jest możliwy do zbudowania, a testy przechodzą
    - Zainstalować wymagania wstępne
    - Zaktualizować środowisko uruchomieniowe lub wykazać, dlaczego stosowany jest *baseline* wprost z huba
    - Dostarczyć `Dockerfile` umożliwiający ponowienie buildu bez konieczności przygotowania/konfiguracji środowiska
    - Dostarczyć `Dockerfile` uruchamiający testy, oparty o poprzedni `Dockerfile` (dziedziczący po nim)
    - Dostarczyć kompozycję *Docker Compose*, uruchamiającą powyższe pliki
- Wykazać, że zbudowany program można uruchomić i wykorzystać
    - W przypadku uruchamiania w kontenerze: wykazać, że funkcjonalność nie jest ograniczana konteneryzacją i opisać, czy dystrybuowanie wybranego oprogramowania w postaci kontenera ma sens
    - W przypadku uruchomienia na zewnątrz kontenera: udokumentować wyciągnięcie artefaktu z kontenera
    - Umotywować wybór (wewnątrz/na zewnątrz)
- Zainstalować w środowisku linuksowym, z wykorzystaniem Dockera, automatyzator *Jenkins*
    - Wykorzystać instrukcję https://www.jenkins.io/doc/book/installing/docker/
    - Zapewnić działanie składnika DIND
    - Przygotować *Blueocean*, załączyć `Dockerfile` tworzący *Blueocean*
- Uruchomić przygotowaną kompozycję *Docker Compose* przy użyciu projektu w Jenkinsie

    - Załączyć logi, umieścić w *commicie* w sposób uniemożliwiający odczytanie szczegółów środowiska (*opsec*) ale łatwy do rozłożenia na czysty tekst
    - Bonus: załączyć zbudowany program jako *„Build Artifact”* projektu
- Przekonwertować projekt na *pipeline* o następujących krokach:
    - Build
    - Test
    - Deploy
- Build oraz Test mają wykonywać kroki kompozycji i opierać się na Dockerfile'ach z poprzednich punktów.
- Skuteczne przejście testów i wdrożenia powinno wygenerować artefakt z builda i umożliwić jego pobranie jako „wypromowana” wersja. Jeżeli nie w postaci jednoznacznego i łatwego do pobrania pliku – konieczne jest opisanie sposobu otrzymania rezultatu („*deliverable*”)
- Decyzja o tym, co jest artefaktem i jak wydobyć go ze skutecznie zakończonej kampanii musi być opisana i umotywowana.
- Kontener ze zbudowaną aplikacją uruchomić na wybranej implementacji Kubernetesa (np. `minikube`). Opisać kroki potrzebne do konfiguracji środowiska i obraną politykę zabezpieczenia hosta

Kryteria zaliczenia

15%	Pozwala odtworzyć kroki i jego zgodne z formatem MD
25%	Dockerfile dla testów i builda
25%	Działający Jenkins i pipeline, ich konfiguracja obecna w sprawozdaniu
20%	Wyprodukowany i dostępny jednoznaczny artefakt
15%	Misc.