Opis Projektu Zaliczeniowego z Cykl życia i narzędzia Devops

Cel projektu: Celem projektu jest wykorzystanie narzędzi DevOps w praktyce poprzez stworzenie i zarządzanie prostą aplikacją webową. Projekt ma obejmować kluczowe aspekty zarządzania kodem źródłowym, konteneryzacji applikacji oraz procesu CI.

Opis projektu

Projekt polega na stworzeniu prostej aplikacji webowej (np. strony internetowej) napisanej w Pythonie, wykorzystującej framework, taki jak Flask. Aplikacja powinna być umieszczona w repozytorium Git, a cały proces wdrożenia powinien być opisany i automatyzowany za pomocą narzędzi DevOps, takich jak GitHub Actions, Jenkins lub GitLab CI/CD. Można wykorzystać inne języki programowania np. Java, C#, Go, Ruby itd. Na podstawie projektu wymagane jest utworzenie fotorelacji w dokumencie pdf.

Wymagania projektowe:

1. **Kod aplikacji**: Prosta aplikacja webowa napisana np. w Pythonie przy użyciu Flask. Aplikacja powinna zawierać minimum jedną stronę, wyświetlającą przykładowe dane.
2. **Konteneryzacja**: Applikacja ma mieć utworzony Dockerfile. Na swoich prywatnych komputerach ma zostać ona zbudowana, uruchomiona i ma być do niej dostęp z przeglądarki internetowej. **Należy w fotorelacji umieścić dowód dostępności applikacji z przeglądarki internetowej.**
3. **Repozytorium Git**: Aplikacja powinna być umieszczona w repozytorium Git, a proces tworzenia nowych funkcji powinien uwzględniać tworzenie kilku **Pull Requestów (PR) (min 3).** PR powinny być wykorzystywane do integrowania zmian z główną gałęzią.
4. **Automatyzacja CI**:
    - W ramach projektu należy zdefiniować proces CI, który będzie uruchamiał się automatycznie przy każdym PR.
        1. Proces powinien obejmować **budowanie skonteryzowanej aplikacji** oraz krok dla podstawowych testów aplikacji (nie trzeba pisac testów, do tego służą inne przedmioty. Wystarczy zrobic krok**: echo „Test”,** aby pokazać zrozumienie procesu).
    - W ramach projektu należy zdefiniować proces CI, który będzie uruchamiał się automatycznie przy każdym mergu do głównego brancha main.
        1. Proces powinien obejmować: budowanie skonteryzowanej aplikacji, krok dla podstawowych testów aplikacji (nie trzeba pisac testów, do tego służą inne przedmioty. Wystarczy zrobic krok: echo „Test”, aby pokazać zrozumienie procesu) oraz wysyłanie zbudowanego obrazu w poprzednim kroku w Github registry **(będzie to konieczne tylko i wyłącznie kiedy zdążymy zrobić to na zajęciach).**
    - Można użyć narzędzi takich jak GitHub Actions, Jenkins, czy GitLab CI/CD do automatyzacji procesu.
5. **Dodatkowe narzędzie bezpieczeństwa** (opcjonalnie):

- Dla oceny 5.0 należy zaimplementować dodatkowy krok uwzględniający narzędzie do analizy bezpieczeństwa kodu lub obrazów kontenerów. Przykładem takiego narzędzia jest **Trivy**, które pozwala na skanowanie kontenerów pod kątem podatności.
- Krok ten powinien być zautomatyzowany i wykonywać się w ramach CI/CD, generując raport z wynikami skanowania.

Kryteria Oceny:

- **3.0**: Projekt spełnia podstawowe wymagania (prosta aplikacja webowa + Pull Requesty – projetk z zajęć) i zawiera opis procesu CI i jego poszczególnych kroków sposób ogólny.
- **4.0**: Projekt spełnia podstawowe wymagania (prosta aplikacja webowa + Pull Requesty – projetk z zajęć) i zawiera szczegółowy oraz staranny opis procesu CI i jego poszczególnych kroków.
- **5.0**:
  - **Opcja 1**: Projekt spełnia wymagania na ocenę 4.0, a dodatkowo zawiera implementację kroku związanego z bezpieczeństwem (np. wykorzystanie Trivy do analizy podatności). Może być użyte inne narzędzie.
  - **Opcja 2**: Student przygotował inny projekt niż na zajęciach, który również zawiera szczegółowy i staranny opis procesu CI wraz ze wszystkimi procedurami.

Szczegóły techniczne:

- **Repozytorium**: Wszystkie pliki projektu powinny być umieszczone w repozytorium GitHub z commitami dokumentującymi postępy prac.
- **Proces CI**:
  - Skrypt CI powinien zawierać kroki takie jak: uruchamianie testów, budowanie obrazu Docker oraz wysłanie do Github repository (jeżeli przerobimy to na zajęciach).
- **Raport bezpieczeństwa** (opcjonalnie): Wyniki analizy bezpieczeństwa powinny być dodane jako raport w repozytorium lub jako wynik na konsoli w systemie CI/CD.

Przykładowe narzędzia:

- **Kontrola wersji**: Git (GitHub, GitLab, Bitbucket)
- **Automatyzacja CI/CD**: GitHub Actions, Jenkins, GitLab CI/CD
- **Bezpieczeństwo**: Trivy (skanowanie obrazów Docker), SonarQube (analiza kodu)

Przykładowy scenariusz realizacji projektu:

1. Utworzenie repozytorium i struktury katalogów dla projektu w Git.
2. Implementacja prostej strony internetowej w Pythonie (np. przy użyciu Flask).
3. Utowrzenie Dockerfile dla powyższej applikacji.

4. Dodanie plików konfiguracyjnych dla CI i skonfigurowanie GitHub Actions lub innego narzędzia.
5. Wprowadzenie zmian w aplikacji poprzez tworzenie PR, które uruchamiają proces CI.
6. (Opcjonalnie) Dodanie kroku z narzędziem Trivy do skanowania obrazu Docker pod kątem podatności.

Dodatkowe uwagi: Dokumentacja ma być częścią projektu, zawierająca instrukcje dotyczące uruchamiania aplikacji oraz szczegółowy opis działania procesu CI. Proszę o wysłanie repo + relacje pdf w spakowanej paczce .zip/.rar na mojego maila lmikolajczyk@wsei.edu.pl wraz z opisem maila uwzględniający: imię, nazwisko, nr albumu, rok, grupa. Ostateczny termin przyjmowanych prac to: 10 stycznia 2025 roku. Projekty będą omawiane indywidualnie na zajęciach zdalnych dnia 11 stycznia 2025.