

## Program szkolenia:

# Jenkins - Continuous Integration

### Informacje:

<b>Nazwa:</b>	<b>Jenkins - Continuous Integration</b>
<b>Kod:</b>	<b>tools-Jenkins-CI</b>
<b>Kategoria:</b>	Narzędzia
<b>Grupa docelowa:</b>	testerzy developerzy
<b>Czas trwania:</b>	2-3 dni
<b>Forma:</b>	50% wykłady / 50% warsztaty

Poznaj ogólne założenia dotyczące CI realizowanego w kompleksowy i efektywny sposób. Dowiedz się, jak skrócić TTM (time to market), podnieść niezawodność i wyeliminować wielogodzinne wydawanie nowej wersji w weekend. Odkryj możliwość zarządzania infrastrukturą w kodzie i sposoby na uczynienie środowiska Continuous Integration Twoim sprzymierzeńcem.

Celem szkolenia jest:

- Przekazanie informacji o automatyzacji zadań związanych z wytwarzaniem oprogramowania
- Poznanie sposobów na zwiększenie jakości i niezawodności tworzonych rozwiązań
- Zaznajomienie się z ekosystemem niezbędnym do zbudowania kompleksowego środowiska CI
- Przedstawienie Continuous Delivery jako następnego kroku po CI

**Uwaga.** Dla firm i zespołów chcących wprowadzić u siebie zarządzalne i skalowalne rozwiązanie dla setek /tysięcy jobów istnieje możliwość wydłużenia szkolenia do 3 dni i przeprowadzenia rozszerzonego wariantu zarządzania Jenkinsem w kodzie z wykorzystaniem Jenkins Job DSL - szczegółowy program:

[Jenkins as code - automatyzacja tworzenia jobów i pipeline'ów w kodzie z wykorzystaniem Job DSL](#)

### Zalety szkolenia:

- Automatyzacja zadań
- Poprawienie jakości oprogramowania
- Skrócenie czasu potrzebnego na wdrożenie kolejnej wersji (TTM)
- Ćwiczenia praktyczne utrwalające przekazaną wiedzę teoretyczną
- Zalecane wzorce i praktyki

## Szczegółowy program:

### 1. Ciągła integracja - wprowadzenie

1.1. cechy

1.2. korzyści

1.3. CI jako składowa procesu wytwarzania oprogramowania w firmie

### 2. Automatyizacja

2.1. dlaczego warto

2.2. popularne problemy przy migracji z procesu ręcznego

2.2.1. sposoby radzenia sobie z nimi

### 3. Serwer ciągłej integracji - wprowadzenie

3.1. funkcje i zadania

3.2. architektura/klasy rozwiązań - wady/zalety, rekomendowane przeznaczenie

3.2.1. proste, często hostowane zewnętrznie - jak Travis

3.2.2. złożone, najczęściej hostowane w firmie - jak Jenkins

### 4. Jenkins - serwer CI

4.1. architektura i kluczowe komponenty

4.2. mechanizmy konfiguracyjne

4.3. joby i widoki

4.4. budowanie typowego projektu

4.5. rozszerzanie możliwości poprzez pluginy

4.6. zarządzanie jobami i widokami w kodzie (wydzielone jako osobny punkt niżej)

4.7. archiwizacja (backup)

4.8. oskryptowywanie zadań administracyjnych

### 5. Automatyczne testowanie kodu - kluczowy element CI

5.1. testowanie automatyczne - potrzeba

5.2. testowanie jednostkowe

5.3. testowanie integracyjne

5.4. testowanie funkcjonalne/akceptacyjne

5.5. testowanie wydajnościowe

5.6. integracja narzędzi do testowania z Jenkinsem

## 6. Satelickie narzędzia i procesy

6.1. repozytorium kodu i praca z kodem

6.1.1. Praca na branchach - workflow

6.1.2. Przeglądu kodu (pre- i post-commit)

6.1.3. Pull requesty

6.1.4. Wsparcie narzędzi

6.2. repozytorium artefaktów

6.3. mechanizm budowania projektu (Maven, Gradle, ...)

6.4. automatyczne testowanie kodu - wydzielone jako osobny punkt

6.5. badanie jakości kodu

6.5.1. Metryki kodu

6.5.2. Statyczna analiza jakości kodu (Checkstyle, PMD, ...)

6.5.3. SonarQube

6.5.4. integracja z Jenkinsem

6.6. wersjonowanie i zarządzanie wydaniem

6.7. zarządzanie zmianami w bazie danych

6.8. zarządzanie infrastrukturą

6.9. automatyzacja wdrożeń

6.10. monitoring i alerting

6.11. baza wiedzy

6.12. issue tracking

## 7. Jenkins as code - zarządzanie Jenkinsem w kodzie z Job DSL (wariant podstawowy lub rozszerzony)

7.1. zarządzanie w kodzie - potrzeba 21 wieku

7.1.1. skalowalność

7.1.2. niezawodność

7.1.3. bezpieczeństwo

7.2. Jenkins Job DSL

7.2.1. architektura i narzędzia

7.2.2. wprowadzenie do Groovy - języka pisania definicji zadań w Jenkinisie

7.2.3. tworzenie jobów i widoków

7.2.4. rozszerzanie możliwości

7.3. migracja istniejących rozwiązań

7.4. Delivery Pipeline - od commitu do wdrożenia

7.4.1. orkiestracja jobów

7.4.2. wizualizacja procesu w Jenkinisie

7.4.3. przykładowe podejście do Delivery Pipeline dla microserwisów

7.5. Automatyczne testowanie

7.6. Jenkins Pipeline - alternatywa do Job DSL

7.6.1. rekomendowane zastosowania

7.6.2. ograniczenia

## 8. Przydatne praktyki

8.1. wzorce i anty-wzorce

8.2. wizualizacja procesu

8.3. zaangażowanie całego zespołu

8.4. rozproszone budowania

8.4.1. skalowalność przy tysiącach jobów

8.5. współbieżność

8.6. integracja z systemami (i procesami) zewnętrznymi

8.7. główne trudności i stosowane rozwiązania

## 9. Continuous Delivery - wprowadzenie

9.1. czym jest?

9.2. co daje?

9.3. kiedy warto stosować?

9.4. wymagania wstępne dotyczące projektu

9.5. przydatne techniki i narzędzia