Kurs DevOps

Lista 12

21, 22, 28, 29 stycznia 2026

Zadanie 1. (14 p.)
Z użyciem Jenkinsa i narzędzi poznanych w trakcie zajęć przygotuj środowisko CI/CD wykonujące weryfikację i wdrożenie wybranej przez Ciebie przykładowej aplikacji.

- W zdalnym repozytorium gita (np. na Githubie) umieść kod źródłowy aplikacji.
- Skonfiguruj lokalną instancję Jenkinsa, tak by periodycznie odpytywała się zdalnego repozytorium,[1] czy został dostarczony nowy komit (opcja GITScm pool lub też w polskiej wersji "Pobierz z repozytorium kodu").[2]
- W Jenkinsie skonfiguruj używanie pliku Jenkinsfile z repozytorium jako definicji potoku.
- Zdefiniuj budowanie i weryfikacje aplikacji w pliku Jenkinsfile.
- Przygotuj obraz dockera, w którym znajdą się zależności potrzebne do zbudowania i zweryfikowania aplikacji oraz agent Jenkinsa.
- Z użyciem Kubernetesa uruchom przygotowany obraz i znajdującego się w nim agenta Jenkinsa.
- Skonfiguruj kontroler Jenkinsa, tak by widział uruchomionego agenta.
- Pokaż, że Jenkins wykrywa zmiany w repozytorium i wywołuje na nich budowanie.
- Zbudowaną przez Jenkinsa aplikację zapisz na zdalnym serwerze (np. lokalnie uruchomiony `miniserve`).
- Zdefiniuj nowy potok w Jenkinsie, który będzie wdrażał aplikację (np. kopiując ją do kontenera, tworząc nowy obraz dockerowy).
- Po zakończeniu testów w potoku weryfikującym bez wykrycia błędów, wywołaj automatycznie potok wdrażający.
- Wystaw z Jenkinsa metryki dla Prometeusza[3] i na tej podstawie przygotuj przykładową tablicę nawigacyjną w Grafanie.

Punktacja:

- Poolowanie zmian ze zdalnego repozytorium: 1p.
- Potok budujący i weryfikujący: 4p.
- Podłączenie agenta: 2p.
- Potok wdrażający: 4p.
- Prometeusz + Grafana: 3p.

[1] Poza periodycznym odpytywaniem się, jest możliwość zdefiniowania hooków w Git/GitHub, które automatycznie wywoływałyby Jenkinsa. Jednakże, biorąc pod uwagę lokalność naszej instancji Jenkinsa, poolowanie będzie prostsze do skonfigurowania.

[2] W przypadku Githuba będzie zapewne potrzebne wygenerowanie tokenu i zapisanie go w Jenkinsie jako sekret. U mnie token drobnoziarnisty zapisany jako "Username with password" był wystarczający.

[3] np. z użyciem `https://plugins.jenkins.io/prometheus/`

Zadanie 2.
Jakie błędy popełniono w:
a)

```
pipeline {
    agent any
    environment {
        EXAMPLE_CREDS = credentials('example-credentials-id')
    }
    stages {
        stage('Example') {
            steps {
                sh("curl -u ${EXAMPLE_CREDS_USR}:${EXAMPLE_CREDS_PSW} https://example.com/")
            }
        }
    }
}
```

b)

```
pipeline {
  agent any
  parameters {
    string(name: 'STATEMENT', defaultValue: 'hello; ls /', description: 'What should I say?')
  }
  stages {
    stage('Example') {
      steps {
        sh("echo ${STATEMENT}")
      }
    }
  }
}
```

Zadanie 3. (2 p.)
Przy pomocy zasobów udostępnianych przez Github w darmowym pakiecie, pokaż jak zdefiniować potok weryfikacyjny z użyciem Github Actions, który:

- Skompiluje i uruchomi prosty program napisany w C++.
- Wykona się dwa razy. Raz na maszynie z procesorem w architekturze x86, a raz z ARM64.
- W celu zdefiniowania maszyny wykorzysta strategię `matrix`.

Zadanie 4. (3 p.)
Zademonstruj jak z użyciem Github Actions wygenerować[4] stronę internetową, którą następnie będzie można hostować z użyciem Github Pages.

[4] np. z użyciem mdBook lub Sphinx