

# Testowanie i wdrażanie systemów informatycznych

## Laboratorium - Jenkins

1. Proszę włączyć aplikację VirtualBox i zaimportować maszynę wirtualną linku podanego przed zajęciami . Przy instalacji lub później w ustawieniach sieciowych wirtualnej maszyny należy zmienić tryb BRIDGE na NAT oraz klikając w *Przekierowanie portów* dokonać nowego wpisu:
  - IP hosta 127.0.0.1
  - port hosta 8443
  - IP gościa 10.0.2.15 (lub ewentualnie inny jaki pojawi się po uruchomieniu maszyny)
  - port gościa 443
2. W przeglądarce internetowej proszę wejść pod adres <https://127.0.0.1:8443> , zaakceptować certyfikat i zalogować się jako *user* z hasłem *t1rhPfqh5IWx* .
3. Po zalogowaniu do GitHuba proszę wejść na repozytorium *mjochab/testowanko* i wykonać fork projektu na swoje konto.
4. Nowo powstały projekt proszę sklonować w NetBeans.
5. W Jenkinsie proszę wybrać *Nowe zadanie/Ogólny* projekt:
  1. w *General* wybrać *Github project* i podać odpowiedni URL projektu na swoim GitHubie
  2. w *Repozytorium kodu* także wpisać ten sam URL,
  3. w *Wyzwalacze zadania - Pobierz z repozytorium kodu* oraz wpisać \* \* \* \* \* (będziemy sprawdzać co minutę czy repozytorium nie zostało uaktualnione; w normalnych warunkach jest to o wiele za często i może spowodować zbanowanie - najlepiej skorzystać z opcji „*GitHub hook trigger for GITScm polling*”, której uruchomienie trwało by zbyt długo aby zrobić to na laboratorium).
  4. w *Budowanie - Dodaj krok budowania / Invoke Ant , Zaawansowane* , w polu *Build file* wpisujemy ścieżkę do skryptu budowania projektu - w tym wypadku:  
*Tescik/build.xml*
6. Wybrać *Uruchom* i zaobserwować wynik w *Logi konsoli*.
7. Proszę wejść do NetBeans, poprawić odpowiedni test i wykonać *commit*.
8. Proszę zaobserwować czy Jenkins wykrył nową wersję projektu i czy udało się ją poprawnie zbudować (skompilować i przetestować).
9. Jeżeli projekt działa spróbujemy zdefiniować nowe zadanie, które będzie do testów używać maszyn wirtualnych z różnymi wersjami Javy. Tym razem użyjemy do tego celu skryptu:

- a. Proszę stworzyć nowy projekt typu *Pipeline*
- b. W *Pipeline/ script* wpisać:

```
pipeline {
  agent any
  stages {
    stage('inicjalizacja') {
      agent any
      steps {
        git 'GITHUB_URL'
      }
    }
    stage('Build') {
      parallel {
        stage('default Java') {
          agent any
          steps {
            sh 'javac -version'
            sh 'ant -f Tescik/build.xml clean'
            sh 'ant -f Tescik/build.xml'
          }
        }
        stage('Java7') {
          agent { docker{
            image "mjochab/jdk-ant-junit:7"
            alwaysPull true
          }
        }
        steps {
          sh 'javac -version'
          sh 'ant -f Tescik/build.xml clean'
          sh 'ant -f Tescik/build.xml'
        }
      }
    }
  }
  post {
    always {
      deleteDir()
    }
  }
}
```

- c. Uruchomić zadanie, obejrzeć logi i zaobserwować różnice po pierwszym i kolejnymi uruchomieniami.
- d. Proszę sprawdzić zachowanie się zadania po odkomentowaniu w teście:  
`//assertTrue(JAVA_7_OR_EARLIER);`
- e. Proszę spróbować analogicznie dodać test dla Javy 8

10. Obrazy Dockera zostały wygenerowane na podstawie prostego skryptu Dockerfile:

```
FROM openjdk:7
```

```
RUN apt-get update && apt-get install -y \  
    ant \  
    junit4 \  
    ant-optional \  
    && rm -rf /var/lib/docker images
```

za pomocą komendy:

```
docker build -t jdk-ant-junit:7 .
```

i opublikowane na [hub.docker.com](https://hub.docker.com) poprzez:

```
docker push mjochab/openjdk:7
```

Skrypt bierze publicznie dostępny obraz Javy i dokłada do niego potrzebne nam zmiany, mianowicie doinstalowuje Apache Ant i JUnit4. Dla ćwiczenia można spróbować wykonać samodzielnie w domu obraz dowolnej innej wersji.

11. Proszę sprawdzić czy podobne zadania jesteście Państwo w stanie łatwiej wykonać korzystając z nowego interfejsu użytkownika Blue Ocean.