

Wstęp

- Budowanym programem jest prosty program napisany w języku C++.

Składa się on z pliku bib.h

```
C bib.h
C bib.h > ...
1  #include <math.h>
2
3  double squareRoot(const double a)
4  {
5      double b = sqrt(a);
6      if(b != b)
7          return -1.0;
8      else
9          return sqrt(a);
10 }
```

Oraz pliku main.cpp

```
G main.cpp
G main.cpp > ...
1  #include<iostream>
2  #include"bib.h"
3  using namespace std;
4  int main()
5  {
6      cout<<squareRoot(24)<<endl;
7      return 0;
8  }
```

- Program kompilowany jest za pomocą Cmake

```
M CMakeLists.txt M
M CMakeLists.txt
1  cmake_minimum_required(VERSION 2.6)
2
3  find_package(GTest REQUIRED)
4  include_directories(${GTEST_INCLUDE_DIRS})
5
6  add_executable(main main.cpp)
7  add_executable(runTests tests.cpp)
8  target_link_libraries(runTests ${GTEST_LIBRARIES} pthread)
9
```

- Program testowany jest za pomocą Google Test.

```

tests.cpp x
tests.cpp > TEST(SquareRootTest, NegativeNos)
1 // tests.cpp
2 #include "bib.h"
3 #include <gtest/gtest.h>
4
5 TEST(SquareRootTest, PositiveNos) {
6     ASSERT_EQ(6, squareRoot(36.0));
7     ASSERT_EQ(18.0, squareRoot(324.0));
8     ASSERT_EQ(25.4, squareRoot(645.16));
9     ASSERT_EQ(0, squareRoot(0.0));
10 }
11
12 TEST(SquareRootTest, NegativeNos) {
13     ASSERT_EQ(-1.0, squareRoot(-15.0));
14     ASSERT_EQ(-1.0, squareRoot(-0.2));
15 }
16
17 int main(int argc, char **argv) {
18     testing::InitGoogleTest(&argc, argv);
19     return RUN_ALL_TESTS();
20 }

```

- Program przeznaczony jest do systemu Linux Ubuntu.
- Założonym artefaktem, który ma zostać uzyskany jest plik .deb służący do instalacji programu.
- Założoną lokalizacją, w której ma znaleźć się zainstalowany program jest /home/user/hell/

Podjęte Kroki

- Utworzono obraz "carmellino/ubuntu" za pomocą dockerfile'a:

```

dockerfile1 A ●
home > moom > moje > devops > 1 > MDO20
1 FROM ubuntu:latest
2 RUN apt-get update -y
3 RUN apt-get install git -y

```

Obraz ten będzie odpowiedzialny za pobranie programu, dockerfile'a instalującego dependencje potrzebne do zbudowania programu oraz pliku control.

- Połączono dockera z dockerhubem za pomocą komendy:

```
sudo docker login -u carmellino
```

A następnie Przesłano utworzony obraz na dockerhub za pomocą komendy:


```
docker push carmellino/ubuntu
```

carmellino / ubuntu

This repository does not have a description 

 Last pushed: 7 days ago

Tags and Scans

 VULNERABILITY SCANNING - DISABLED [Enable](#)

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
 latest		---	7 days ago

[See all](#)

- Utworzono dockerfile'a o treści:

```
1 FROM ubuntu:latest
2 RUN apt-get update -y
3 RUN apt-get install git -y
4 ENV TZ=Europe/Warsaw
5 RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
6 RUN apt-get install cmake -y
7 RUN apt-get install g++ -y
8 RUN apt-get install libgtest-dev -y
9 WORKDIR "/usr/src/gtest"
10 RUN cmake CMakeLists.txt
11 RUN make
12 RUN cp lib/libgtest.a /usr/lib
13 RUN cp lib/libgtest_main.a /usr/lib
14 WORKDIR "/"
```

Następnie przesłano go do repozytorium programu.

Dockerfile ten odpowiedzialny będzie za stworzenie środowiska do zbudowania programu.

- Utworzono plik control o treści:

```
1 Package: hell
2 Version: 0.2
3 Maintainer: King Foo
4 Architecture: all
5 Description: hellhellhell
```

Następnie przesłano go do repozytorium programu.

Plik control jest potrzebny do zbudowania pliku .deb.

- Utworzono nowy pipeline w Jenkinsie ze skryptem:

```

pipeline {
  agent none
  stages
  {
    stage('First Clean')
    {
      agent any
      steps
      {
        cleanWs()
      }
    }
    stage('Git Gud')
    {
      agent
      {
        docker
        {
          image 'carmellino/ubuntu'
          args '-u root:sudo'
        }
      }
      steps
      {
        sh 'ls'
        sh 'git clone https://github.com/carmellino/naDevOps.git'
        sh 'mv naDevOps/Dockerfile Dockerfile'
      }
    }
    stage('Dockerfile Build')
    {
      agent {dockerfile{filename 'Dockerfile' args '-u root:sudo'}}
      steps
      {
        sh 'mkdir build'
        sh 'cp naDevOps/* build'
        sh '(cd build;cmake CMakeLists.txt;make;)'
      }
    }
    stage('Build Tests')
    {
      agent any
      steps
      {
        sh '(cd build;./runTests)'
      }
    }
    stage('Publish')
    {
      agent {docker {image 'ubuntu'
      args '-u root:sudo'}}
      steps
      {
        sh 'mkdir hell && mkdir hell/DEBIAN'
        sh 'mkdir hell/home'
        sh 'mkdir hell/home/user'
        sh 'mkdir hell/home/user/hell'
        sh 'cp build/main hell/home/user/hell/hell'
        sh 'cp naDevOps/control hell/DEBIAN/control'
        sh 'dpkg-deb --build hell'
        archiveArtifacts artifacts: 'hell.deb', fingerprint: true
      }
    }
  }
}

```

- Stage "First Clean" odpowiedzialny jest za czyszczenie workspace'a przed rozpoczęciem budowania.
- Stage "Git Gud" odpowiedzialny jest za sklonowanie repozytorium zawierającego program, dockerfile'a oraz pliku control.
- Stage "Dockerfile Build" tworzy kontener ze środowiskiem wymaganym do zbudowania programu.
- Stage "Build Test" odpowiedzialny jest za testowanie działania zbudowanego programu.
- Stage "Publish" tworzy plik .deb, a następnie zapisuje go jako artefakt.

S	W	Name	Last Success	Last Failure	Last Duration
✓	🔗	hell	2 days 0 hr #25	2 days 1 hr #23	20 sec

- Zbudowano pipeline:

Console Output

```

Started by user Diego Vega
[Pipeline] Start of Pipeline
[Pipeline] stage
[Pipeline] { (First Clean)
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/hell
[Pipeline] {
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Git Gud)
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/hell
[Pipeline] {
[Pipeline] isUnix
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ docker inspect -f . carmellino/ubuntu
.

```

Otrzymany artefakt:

Pipeline hell

 **Last Successful Artifacts**
 **hell.deb** 804 B  [view](#)

 **Recent Changes**

Stage View

		First Clean	Git Gud	Dockerfile Build	Build Tests	Publish	Declarative: Post Actions
Average stage times: (Average full run time: ~18s)		235ms	4s	7s	498ms	4s	218ms
#25	May 08 13:25 No Changes	243ms	5s	8s	533ms	5s	
#24	May 08 13:09 No Changes	243ms	4s	7s	486ms	4s	

- Pobrano otrzymany artefakt na system ubuntu i zainstalowano go za pomocą komendy:

`sudo dpkg -i Downloads/hell.deb`

```
moom@moom-GL552VW:~$ sudo dpkg -i Downloads/hell.deb
[sudo] password for moom:
(Reading database ... 236644 files and directories currently installed.)
Preparing to unpack Downloads/hell.deb ...
Unpacking hell (0.2) over (0.2) ...
Setting up hell (0.2) ...
moom@moom-GL552VW:~$
```

- Sprawdzono poprawną instalację programu:

```
moom@moom-GL552VW:/home$ cd user/hell/
moom@moom-GL552VW:/home/user/hell$ ./hell
4.89898
moom@moom-GL552VW:/home/user/hell$
```

Program został zainstalowany w założonej lokalizacji i działa poprawnie.