
	Uniwersytet Technologiczno-Przyrodniczy im. J. i J. Śniadeckich w Bydgoszczy Wydział Telekomunikacji, Informatyki i Elektrotechniki	
Przedmiot	Narzędzia programistyczne	
Prowadzący	dr inż. Damian Ledziński	
Temat projektu	Środowisko programistyczne	
Student	Łukasz Farulewski	
Kierunek	Informatyka Stosowana	
Data wykonania	19.06.2019	

Celem projektu jest przygotowanie zautomatyzowanego środowiska programistycznego. Począwszy od budowy projektu (kompilacji), przeprowadzenie testów jednostkowych, pakowanie aplikacji do pliku .jar, budowanie dockerowych obrazów (Docker Hub oraz Google Cloud Registry) po przesyłanie obrazów aplikacji do repozytorium Docker Hub i uruchomienie kontenera aplikacji na dedykowanym serwerze. Dodatkowo zadaniem projektu jest komunikacja z bazą danych i możliwość zapisu informacji do bazy.

W projekcie wykorzystano aplikację webową w technologii **Spring Boot** z silnikiem szablonów **Thymeleaf**. Jest to projekt **Maven**, który posłużył do budowy i testowania programu. Aplikacja została skonfigurowana z bazą danych H2. **Baza H2** jest wbudowana do pamięci RAM. Jest ona ulotna co oznacza, że po zakończeniu sesji traci się wprowadzone wcześniej dane.

Jako repozytorium dla aplikacji wykorzystano **GitHub**, który dodatkowo może posłużyć jako system kontroli wersji.

Najważniejszą częścią projektu jest **Jenkins** i jego konfiguracja. Jest to serwer CI/CD, który służy do ciągłej integracji projektów, ich rozwoju i deploymentu. Jenkinsa można skonfigurować w dwojaki sposób – poprzez Jenkinsfile – plik konfiguracyjny oraz deklaratywnie w GUI Jenkinsa. W projekcie wykorzystano obie metody.

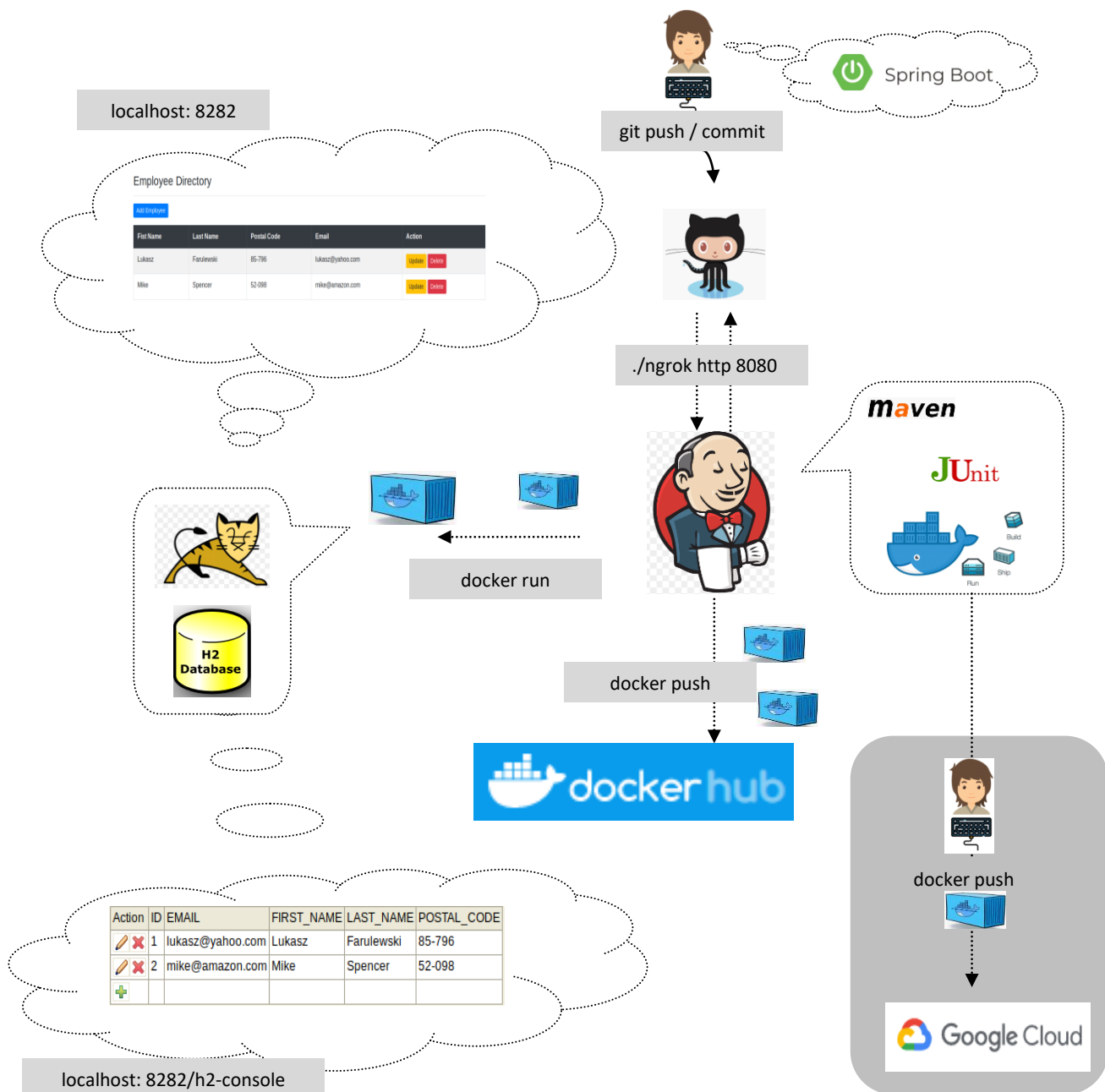
Aby Jenkins mógł komunikować się z repozytorium na GitHub należało ustawić publiczny tunel **ngrok** dla lokalnego hosta 8080 na którym pracuje serwer CI/CD Jenkins.

W projekcie aplikacja webowa została spakowana do kontenera **Docker** przy pomocy pliku Dockerfile. Kontener taki poza aplikacją zawiera całe środowisko uruchomieniowe wraz z bibliotekami i można je uruchomić na dowolnym systemie operacyjnym. Przy tym jest znacznie lżejszą technologią w porównaniu do VM, ponieważ w przeciwieństwie do VM kontener nie emuluje całego komputera i nie posiada wydzielonego SO.

Aplikacja została uruchomiona na serwerze **Tomcat**. Jest to kontener serwletów służący do uruchamiania aplikacji webowych i Java EE. Ponieważ domyślnie Tomcat pracuje na porcie 8080, pracę serwera przekierowano na port 9000. Ten port umożliwi pracę aplikacji na lokalnym hoście. Aby jednak uruchomić aplikację z kontenera należało ten port zmapować na port 8282, który ostatecznie posłużył do uruchomienia aplikacji z kontenera Docker.

W projekcie wykorzystano także repozytorium Google Cloud (Google Cloud Registry). Ponieważ jednak usługi Google są płatne, GCR posłużył do przetestowania repozytorium na Google i umieszczania tam wybranych obrazów dockerowych manualnie poprzez polecenie docker push.

Poniżej przedstawiono pipeline projektu.



java , maven

```
root@lfarul:/home/lfarul/Pulpit# java -version
openjdk version "11.0.3" 2019-04-16
OpenJDK Runtime Environment (build 11.0.3+7-Ubuntu-1ubuntu218.04.1)
OpenJDK 64-Bit Server VM (build 11.0.3+7-Ubuntu-1ubuntu218.04.1, mixed mode)
root@lfarul:/home/lfarul/Pulpit# mvn --version
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 11.0.3, vendor: Oracle Corporation, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: pl_PL, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-51-generic", arch: "amd64", family: "unix"
root@lfarul:/home/lfarul/Pulpit# █
```

docker

```
root@lfarul:/home/lfarul/Pulpit# docker version
Client:
 Version:           18.09.6
 API version:       1.39
 Go version:        go1.10.8
 Git commit:        481bc77
 Built:             Sat May  4 02:35:57 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:           18.09.6
  API version:       1.39 (minimum version 1.12)
  Go version:        go1.10.8
  Git commit:        481bc77
  Built:             Sat May  4 01:59:36 2019
  OS/Arch:           linux/amd64
  Experimental:      false
root@lfarul:/home/lfarul/Pulpit# █
```

Jenkins

```
root@lfarul:/home/lfarul/Pulpit# systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; generated)
   Active: active (exited) since Wed 2019-06-19 10:51:30 CEST; 4h 17min ago
     Docs: man:systemd-sysv-generator(8)
    Process: 1590 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCESS)

cze 19 10:51:09 lfarul systemd[1]: Starting LSB: Start Jenkins at boot time...
cze 19 10:51:25 lfarul jenkins[1590]: Correct java version found
cze 19 10:51:25 lfarul jenkins[1590]: * Starting Jenkins Automation Server jenkins
cze 19 10:51:27 lfarul su[2013]: Successful su for jenkins by root
cze 19 10:51:27 lfarul su[2013]: + ??? root:jenkins
cze 19 10:51:27 lfarul su[2013]: pam_unix(su:session): session opened for user jenkins by (uid=0)
cze 19 10:51:30 lfarul jenkins[1590]: ...done.
cze 19 10:51:30 lfarul systemd[1]: Started LSB: Start Jenkins at boot time.
root@lfarul:/home/lfarul/Pulpit# █
```

```
lfarul@lfarul:~/Pulpit$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/

lfarul@lfarul:~/Pulpit$
```

Instalacja Jenkinsa

```
lfarul@lfarul:~/Pulpit$ sudo apt-get install jenkins
Czytanie list pakietów... Gotowe
Budowanie drzewa zależności
Odczyt informacji o stanie... Gotowe
The following additional packages will be installed:
  daemon
Zostaną zainstalowane następujące NOWE pakiety:
  daemon jenkins
0 aktualizowanych, 2 nowo instalowanych, 0 usuwanych i 0 nieaktualizowanych.
Konieczne pobranie 73,7 MB archiwów.
Po tej operacji zostanie dodatkowo użyte 76,0 MB miejsca na dysku.
Kontynuować? [T/n] t
Pobieranie:1 http://pl.archive.ubuntu.com/ubuntu/artful/universe/amd64/daemon\_0.6.4-1 [98,2 kB]
Pobieranie:2 https://pkg.jenkins.io/debian-stable/binary/jenkins\_2.138.1 [73,6 MB]
Pobrano 73,7 MB w 2min 4s (592 kB/s)
Wybieranie wcześniej niewybranego pakietu daemon.
(Odczytywanie bazy danych ... 219237 plików i katalogów obecnie zainstalowanych.)
Przygotowywanie do rozpakowania pakietu .../daemon_0.6.4-1_amd64.deb ...
Rozpakowywanie pakietu daemon (0.6.4-1) ...
Wybieranie wcześniej niewybranego pakietu jenkins.
Przygotowywanie do rozpakowania pakietu .../jenkins_2.138.1_all.deb ...
Rozpakowywanie pakietu jenkins (2.138.1) ...
Przetwarzanie wyzwalaczy pakietu ureadahead (0.100.0-20)...
Przetwarzanie wyzwalaczy pakietu systemd (234-2ubuntu12.4)...
Przetwarzanie wyzwalaczy pakietu man-db (2.7.6.1-2)...
Konfigurowanie pakietu daemon (0.6.4-1) ...
Konfigurowanie pakietu jenkins (2.138.1) ...
Przetwarzanie wyzwalaczy pakietu systemd (234-2ubuntu12.4)...
Przetwarzanie wyzwalaczy pakietu ureadahead (0.100.0-20)...
lfarul@lfarul:~/Pulpit$
```

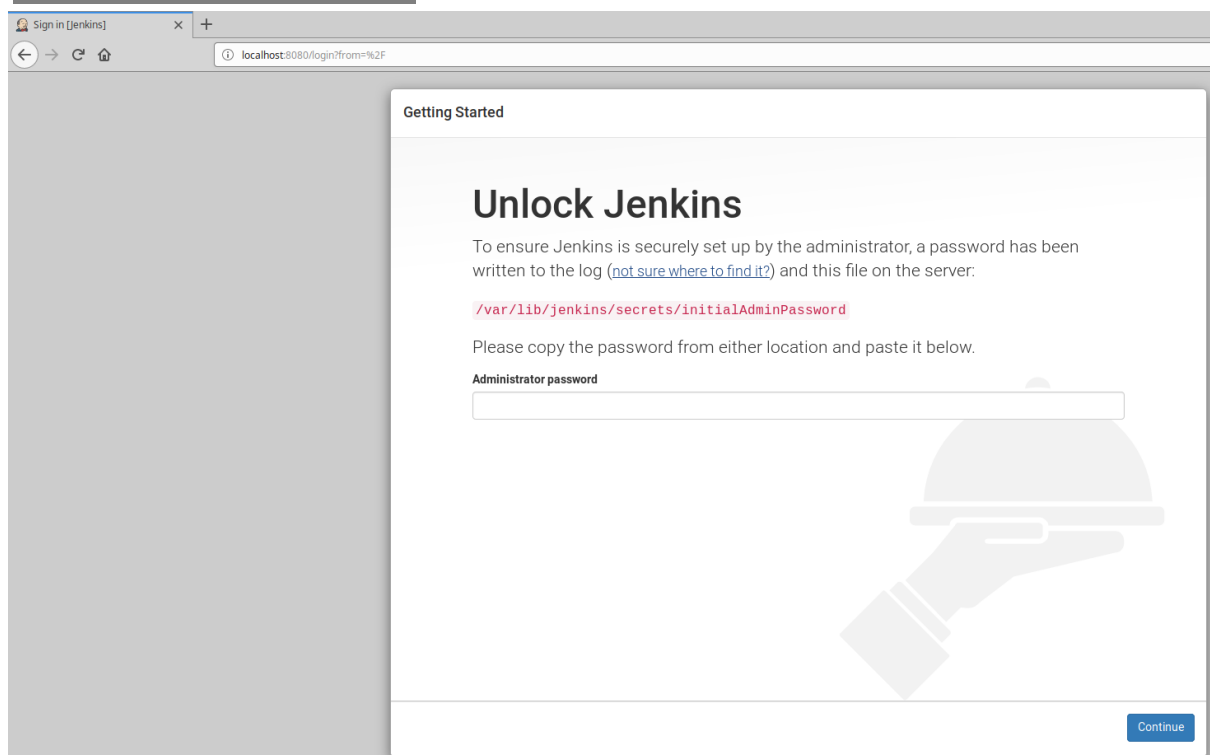
Konfiguracja firewalla

```
lfarul@lfarul:~/Pulpit$ sudo ufw allow 8080
Zaktualizowano reguły
Zaktualizowano reguły (v6)
lfarul@lfarul:~/Pulpit$ sudo ufw status
Stan: nieaktywny
lfarul@lfarul:~/Pulpit$ sudo ufw allow OpenSSH
ERROR: Nie mogę znaleźć profilu pasującego do 'OpenSSH'
lfarul@lfarul:~/Pulpit$ sudo apt-get install ssh
Zainstalowano 0 pakietów, zaktualizowano 0, zniszczono 0,
lfarul@lfarul:~/Pulpit$ sudo ufw enable
Zapora jest aktywowana i włączana podczas startu systemu
lfarul@lfarul:~/Pulpit$ sudo ufw status
Stan: aktywny

Do                          Działanie  Z
--                          -
8080                         ALLOW      Anywhere
OpenSSH                     ALLOW      Anywhere
8080 (v6)                   ALLOW      Anywhere (v6)
OpenSSH (v6)                ALLOW      Anywhere (v6)

lfarul@lfarul:~/Pulpit$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
e70f69fe0d3a41f0b03d866ae3efc952
lfarul@lfarul:~/Pulpit$
```

Ustawianie Jenkinsa



Ustawianie nowego użytkownika

W późniejszym etapie
został dodany nowy
użytkownik - lukasz

SetupWizard [Jenkins] x +

localhost:8080

Dodawanie pierwszego użytkownika

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.138.1

[Kontynuuj jako administrator](#) [Zapisz i zakończ](#)

Dodawanie pierwszego użytkownika

Instance Configuration

Jenkins URL:

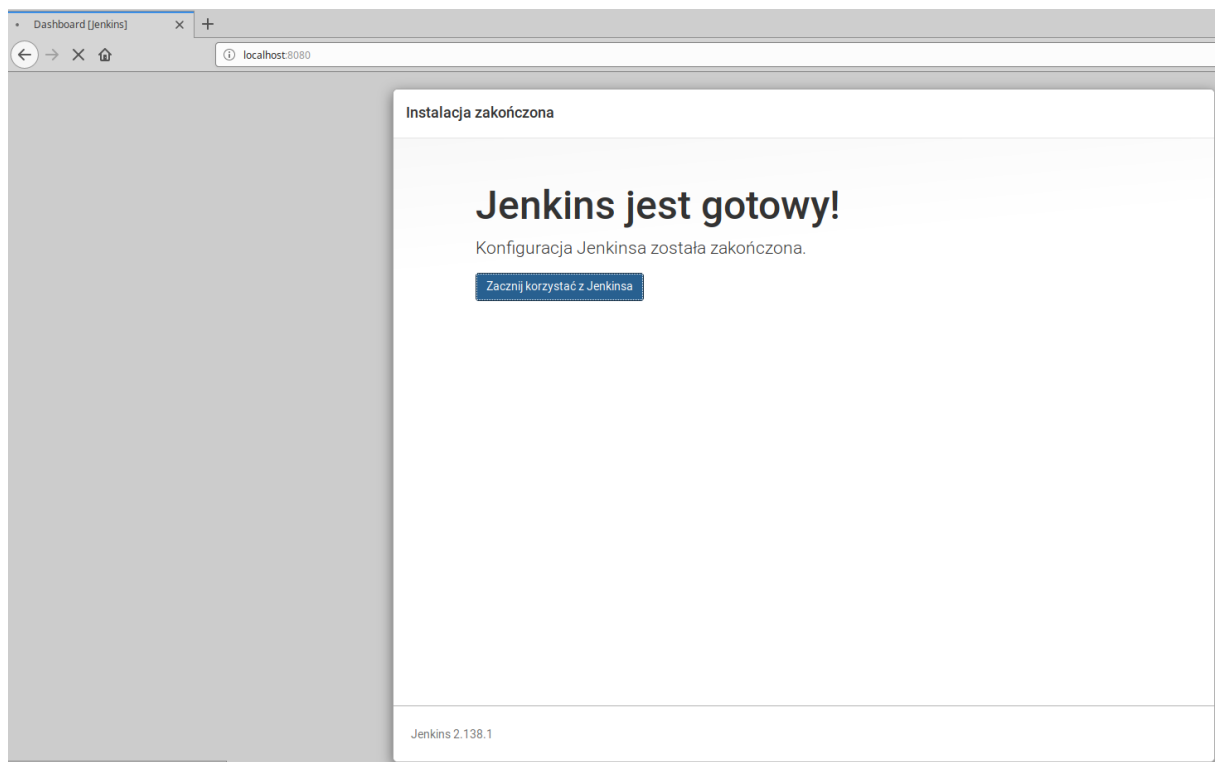
The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.138.1

[Not now](#)

[Save and Finish](#)



Dodawanie użytkownika Jenkinsa do Dockera

```
lfarul@lfarul:~/Pulpit$ sudo usermod -aG docker jenkins
lfarul@lfarul:~/Pulpit$ grep docker /etc/group
docker:x:129:lfarul,jenkins
lfarul@lfarul:~/Pulpit$
```

Generowanie tokena

GitHub, Inc. [US] <https://github.com/settings/tokens>

Search or jump to... Pull requests Issues Marketplace Explore

Settings / Developer settings

OAuth Apps
GitHub Apps
Personal access tokens

Personal access tokens

Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ e5e9b19f16e74b35744649b851ac2a969fab32e [Copy](#) Delete

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Utworzenie repozytorium na GitHub

lfarul / EmployeeRegistryH2

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights Settings

No description, website, or topics provided. [Edit](#)

[Manage topics](#)

111 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find File Clone or download

lfarul Update Jenkinsfile Latest commit 01dc439 8 minutes ago

.mvn/wrapper	-	16 days ago
src	Update application.properties	2 days ago
.gitignore	-	16 days ago
Dockerfile	Update Dockerfile	4 days ago
Jenkinsfile	Update Jenkinsfile	8 minutes ago
README.md	Update README.md	12 days ago
mvnw	-	16 days ago
mvnw.cmd	-	16 days ago
pom.xml	Update pom.xml	5 days ago

W celu komunikacji Jenkinsa oraz GitHub należy stworzyć adres publiczny oraz tunel dla hosta 8080 na którym pracuje Jenkins. Ngrok wygeneruje publiczny adres hosta który będzie przysyłał żądania do lokalnego API.

Instalacja ngrok

```
root@lfarul:/home/lfarul/Pulpit# unzip ngrok-stable-linux-amd64.zip
Archive:  ngrok-stable-linux-amd64.zip
  inflating: ngrok
root@lfarul:/home/lfarul/Pulpit#
```

```
lfarul@lfarul:~/Pulpit$ ngrok --version
ngrok version 2.2.8
```

Wskazuje port
lokalnego hosta, który
chcę upublicznić

```
./ngrok http 8080
```

ngrok by @inconshreveable

```
Session Status      online
Account             lfaryl (Plan: Free)
Version             2.3.29
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding            http://5c704703.ngrok.io -> http://localhost:8080
Forwarding            https://5c704703.ngrok.io -> http://localhost:8080

Connections         ttl    opn    rt1    rt5    p50    p90
                   16     0      0.00   0.00   9.44   14.16
```

Pod tym adresem
został wystawiony
mój lokalny serwer

Explore ngrok

Status

Reserved

Auth

Team

Admin

Billing

Want more from ngrok?

[Upgrade now](#)

Tunnels Online

	URL	Client IP	Region	Established
▶	https://5c704703.ngrok.io	37.30.13.87	us	Jun 19th, 2019 08:53:48 UTC
▶	http://5c704703.ngrok.io	37.30.13.87	us	Jun 19th, 2019 08:53:48 UTC

! Jeżeli użytkownik nie ma założonego płatnego konta ngrok, sesja utworzonego publicznego tunelu kończy się po 7 godzinach.

Po ustawieniu publicznego tunelu **ngrok** dla lokalnego hosta 8080, komunikacja między Jenkinsem i GitHubem umożliwia automatyzację procesu budowania i testowania aplikacji po każdej zmianie i komicie w GitHubie

Teraz Jenkins i GitHub widzą się i mogą ze sobą rozmawiać. Jenkins może po każdym trigerze (np. commit lub push) pobrać aplikację z repozytorium GitHub i zgodnie z plikiem konfiguracyjnym Jenkinsfile przeprowadzić określone stage – np. build, run.

Konfiguracja Webhook po stronie GitHub

Webhook nie zadziała na lokalnym hoście. Dlatego w ustawieniach Payload URL podałem adres publiczny.

lfarul / EmployeeRegistryH2

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Security

Insights

Settings

Options

Collaborators

Branches

Webhooks

Notifications

Integrations & services

Deploy keys

Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <http://5c704703.ngrok.io/github-webhook/> (all events)

Edit

Delete

Projekt krok po kroku.

Jenkinsfile

```
1  pipeline {
2    agent any
3    stages {
4
5      /*stage ("Checkout") {
6        steps {
7          echo "Checking out...."
8          git credentialsId: 'git-creds', url: 'https://github.com/lfarul/EmployeeRegistryH2'
9        }
10     }
11    */
12    // Kompiluje plik
13    stage("Compile / Build") {
14      steps {
15        echo "Compiling / Building..."
16        sh 'mvn compile'
17      }
18    }
19
20    // Przeprowadzam testy jednostkowe
21    stage("JUnit Test") {
22      steps {
23        echo "JUnit testing..."
24        sh 'mvn test'
25      }
26    }
27  }
```

„Checkout” – pobranie aplikacji z repozytorium GitHub. W tym przypadku polecenie zostało wycommentowane, ponieważ Checkout został skonfigurowany deklaratywnie w GUI Jenkinsa.

Kompilacja oraz testy jednostkowe. Jeżeli podczas budowy projektu zostaną wykryte błędy, to już na tym etapie zakończy się pipeline.

```

28 // Pakuje aplikacje do pliku .jar
29 stage("Package") {
30     steps {
31         echo "Packaging..."
32         sh 'mvn package'
33     }
34 }
35
36 /*
37 // Uruchamiam aplikację Jenkinsem z wykorzystaniem mavena
38 stage("Mvn Package & Run") {
39     steps {
40         echo "Packaging and Running..."
41         sh 'mvn package && java -jar target/thymeleaf-demo-0.0.1-SNAPSHOT.jar'
42     }
43 }
44 */
45
46 // Buduje obraz Dockera dla Docker Registry
47 stage("Build Docker image for Docker Hub"){
48     steps{
49         echo "Building Docker image for Docker Registry..."
50         // lfarul to mój username na dockerhub i musi być w nazwie image / nazwa obrazu : wersja obrazu
51         sh 'docker build -t lfarul/employeeeeregistry:17.0 .'
52     }
53 }
54
55 // Robie push obrazu Dockera na chmurę Dockera
56 stage("Push Docker image to Docker Registry"){
57     steps{
58         echo "Pushing Docker image to Docker Registry..."
59         withCredentials([string(credentialsId: 'docker-pwd', variable: 'dockerHubPwd')]) {
60             sh "docker login -u lfarul -p ${dockerHubPwd}"
61         }
62         sh 'docker push lfarul/employeeeeregistry:17.0'
63     }
64 }

```

Tutaj maven pakuje aplikację do pliku jar.

Aplikację można także spakować do pliku jar i następnie uruchomić bez pakowania jej do kontenera.

Buduję obraz dockerowy aplikacji dla Docker Hub.

Obraz zostaje „wypchnięty” na Docker Hub. Tutaj poza poleceniem docker push należało użyć specjalne credentiala wygenerowane w Snippet Generator w GIU Jenkinsa.

```

66 // Buduje obraz Dockera dla Google Cloud
67 stage("Build Docker image for Google Cloud"){
68     steps{
69         echo "Building Docker image for Google Cloud..."
70         sh 'docker build -t gcr.io/nowyprojekt-235718/employeeeeregistry:17.0 .'
71     }
72 }
73
74 // Uruchamiam aplikację w kontenerze na zmapowanym porcie 8282
75 stage("Run Docker container"){
76     steps{
77         echo "Running Docker container"
78         sh 'docker run -d -p 8282:9000 lfarul/employeeeeregistry:17.0'
79     }
80 }
81 }
82 }

```

Buduję obraz dockerowy aplikacji dla Google Cloud Registry.

Uruchamiam aplikację na serwerze na porcie 8282.

Branch: master
EmployeeRegistryH2 / Dockerfile
Find file
Copy path

Ilfarul Update Dockerfile
0e3cfca 4 days ago
1 contributor

8 lines (6 sloc) 219 Bytes
Raw
Blame
History

```

1 FROM openjdk:11-jre-slim
2 COPY ./target/thymeleaf-demo-0.0.1-SNAPSHOT.jar /usr/src/thymeleafdemo/
3 WORKDIR /usr/src/thymeleafdemo
4 EXPOSE 9000
5 ENTRYPOINT ["java", "-jar", "thymeleaf-demo-0.0.1-SNAPSHOT.jar"]
6
7 #alpine:latest

```

W Dockerfile należy zwrócić uwagę jakiej wersji obraz się buduje (FROM), czy będzie on spójny ze środowiskiem, które kompiluje i uruchamia projekt – czyli dockerowy obraz aplikacji.

Wykaz przykładowych obrazów oraz pracujących kontenerów: *docker images* oraz *docker ps*.

```

root@lfarul:/home/lfarul/Pulpit# docker images
REPOSITORY                                TAG                IMAGE ID           CREATED            SIZE
gcr.io/nowyprojekt-235718/employeeeeregistry 17.0               e95ada1b8b3c      7 hours ago       312MB
gcr.io/nowyprojekt-235718/employeeeeregistry 15.0               712d14545440     44 hours ago      312MB
lfarul/employeeeeregistry                   15.0               712d14545440     44 hours ago      312MB
lfarul/employeeeeregistry                   12.0               86af64eb34d0     3 days ago        312MB
gcr.io/nowyprojekt-235718/employeeeeregistry 12.0               86af64eb34d0     3 days ago        312MB
lfarul/employeeeeregistry                   <none>             93682b1bc020     3 days ago        312MB
lfarul/employeeeeregistry                   11.0               81ef642dfd5d     3 days ago        312MB
gcr.io/nowyprojekt-235718/employeeeeregistry 11.0               81ef642dfd5d     3 days ago        312MB
lfarul/employeeeeregistry                   10.0               372353a54322     3 days ago        5.53MB
gcr.io/nowyprojekt-235718/employeeeeregistry 10.0               372353a54322     3 days ago        5.53MB
lfarul/employeeeeregistry                   4.0                48c031bb6048     5 days ago        687MB
gcr.io/nowyprojekt-235718/employeeeeregistry 4.0                48c031bb6048     5 days ago        687MB
lfarul/employeeeeregistry                   <none>             64d4282c33f2     5 days ago        687MB
lfarul/employeeeeregistry                   3.0                953d5b88ac01     5 days ago        687MB
gcr.io/nowyprojekt-235718/employeeeeregistry 3.0                953d5b88ac01     5 days ago        687MB
lfarul/employeeeeregistry                   <none>             38ecccda6581     5 days ago        148MB
openjdk                                     8-jdk-alpine       a3562aa0b991     5 weeks ago       105MB
alpine                                     latest              055936d39205     5 weeks ago       5.53MB
openjdk                                     11-jre-slim        a71b975306d9     6 weeks ago       273MB
hello-world                               latest              fce289e99eb9     5 months ago      1.84kB
java                                       8                   d23bdf5b1b1b     2 years ago       643MB
root@lfarul:/home/lfarul/Pulpit# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
root@lfarul:/home/lfarul/Pulpit#

```

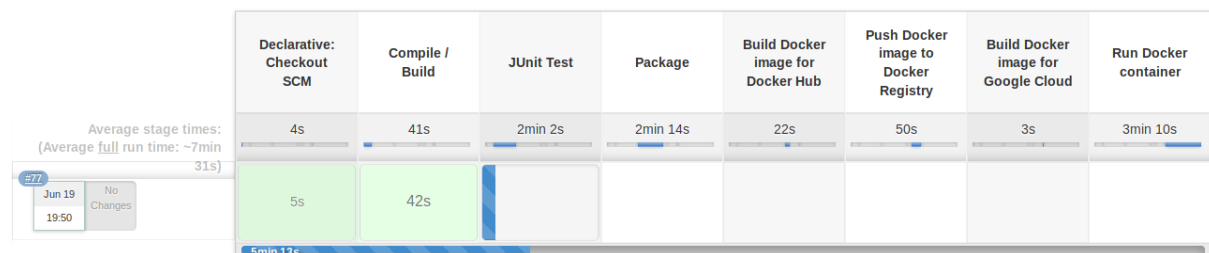
Jak widać obecnie nie pracuje żaden kontener.

Uruchomienie aplikacji – employeeeeregistry:18.0 – Zmiana w pliku konfiguracyjnym Jenkinsfile wyzwała pipeline i uruchamia proces CI/CD.

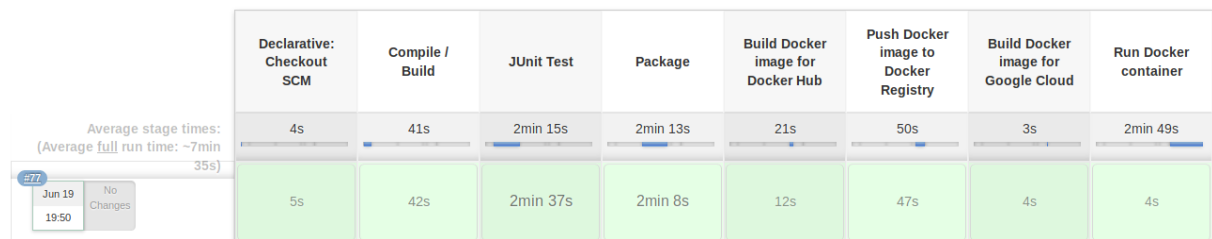
Pipeline EmployeeRegistryH2



Stage View



Stage View



Po zakończeniu procesu powinien zostać utworzony nowy obraz aplikacji w wersji: 18.0 a także na porcie 8282 powinna zostać uruchomiona aplikacja.

\$ docker images

Zostały zbudowane dwa nowe obrazy, każdy o innej konwencji nazewnictwa – jeden przeznaczony do Docker Hub (lfarul) a drugi do Google Cloud Registry (gcr.io)

```
lfarul@lfarul:~/Pulpit$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
gcr.io/nowyprojekt-235718/employeeeeregistry	18.0	3848004b1f01	8 minutes ago	312MB
lfarul/employeeeeregistry	18.0	3848004b1f01	8 minutes ago	312MB
lfarul/employeeeeregistry	17.0	e95ada1b8b3c	9 hours ago	312MB
gcr.io/nowyprojekt-235718/employeeeeregistry	17.0	e95ada1b8b3c	9 hours ago	312MB
lfarul/employeeeeregistry	16.0	25206dfb3086	25 hours ago	312MB
gcr.io/nowyprojekt-235718/employeeeeregistry	16.0	25206dfb3086	25 hours ago	312MB
lfarul/employeeeeregistry	14.0	38c40bb781a2	46 hours ago	312MB
gcr.io/nowyprojekt-235718/employeeeeregistry	14.0	38c40bb781a2	46 hours ago	312MB
lfarul/employeeeeregistry	<none>	6efe6e83b05e	2 days ago	312MB
gcr.io/nowyprojekt-235718/employeeeeregistry	13.0	37948e275bed	2 days ago	312MB
lfarul/employeeeeregistry	13.0	37948e275bed	2 days ago	312MB
lfarul/employeeeeregistry	12.0	a6b968ec61aa	2 days ago	312MB
gcr.io/nowyprojekt-235718/employeeeeregistry	12.0	a6b968ec61aa	2 days ago	312MB
lfarul/employeeeeregistry	<none>	22e0d374975f	2 days ago	312MB

\$ docker ps

Został także uruchomiony kontener z aplikacją w wersji: 18.0.

```
lfarul@lfarul:~/Pulpit$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
925db85920a8	lfarul/employeeeeregistry:18.0	"java -jar thymeleaf..."	9 minutes ago	Up 9 minutes	0.0.0.0:8282->9000/tcp	optimistic_shtern

```
lfarul@lfarul:~/Pulpit$
```

Na porcie 8282 została uruchomiona aplikacja webowa.

localhost:8282/employees/list

Employee Directory

Add Employee

Fist Name	Last Name	Postal Code	Email	Action
-----------	-----------	-------------	-------	--------

Employee Directory

Save Employee

Save

Dodaje nowego pracownika do bazy danych w formularzu aplikacji webowej.

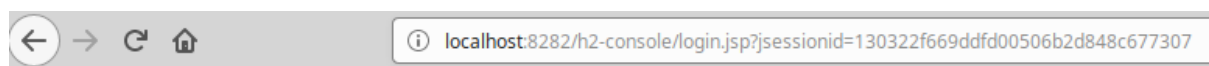
[Back to Employees List](#)

Employee Directory

[Add Employee](#)

Fist Name	Last Name	Postal Code	Email	Action
Lukasz	Farulewski	85-796	lukasz@yahoo.com	<div><div>Update</div><div>Delete</div></div>

Połączenie z bazą dostępne jest na porcie 8282/h2-console



Logowanie do bazy H2 na porcie 8282/h2-console

English ▾ Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded) ▾

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:testdb

User Name: sa

Password:

Connect Test Connection

jdbc:h2:mem:testdb

- EMPLOYEE
- INFORMATION_SCHEMA
- Sequences
- Users

H2 1.4.199 (2019-03-13)

Nowy pracownik jest widoczny w bazie danych.

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM EMPLOYEE

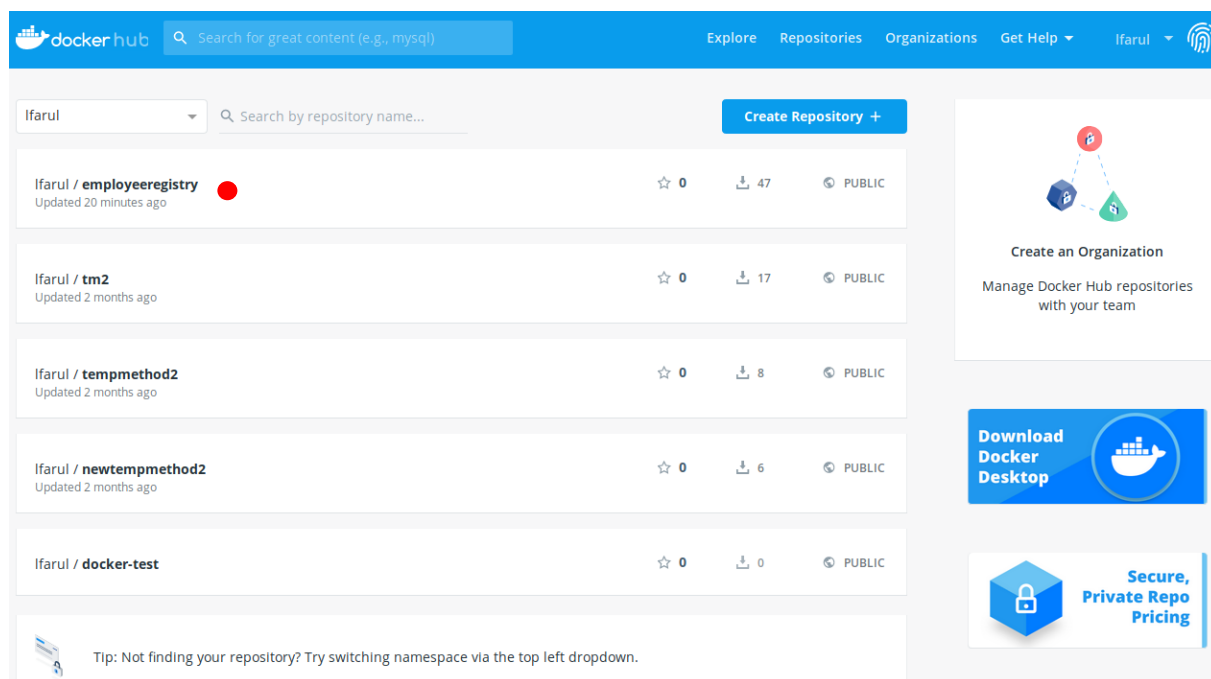
SELECT * FROM EMPLOYEE;

ID	EMAIL	FIRST_NAME	LAST_NAME	POSTAL_CODE
1	lukasz@yahoo.com	Lukasz	Farulewski	85-796

(1 row, 47 ms)

Edit

Obraz aplikacji znajduje się na chmurze Dockera – Docker Hub.

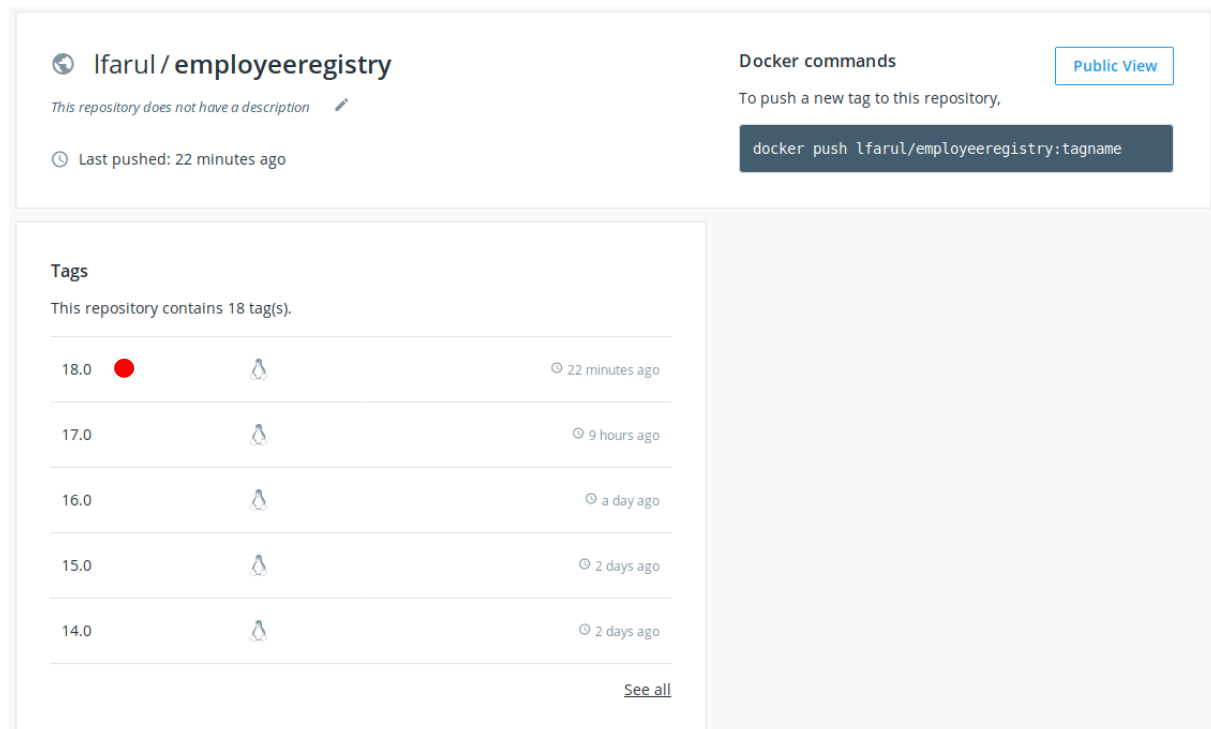


The screenshot shows the Docker Hub interface. At the top, there's a navigation bar with the Docker Hub logo, a search bar, and links for Explore, Repositories, Organizations, Get Help, and the user 'lfarul'. Below the navigation bar, there's a section for the user 'lfarul' with a search bar and a 'Create Repository +' button. A list of repositories is displayed:

Repository	Stars	Downloads	Visibility
lfarul / employeeeegistry Updated 20 minutes ago	0	47	PUBLIC
lfarul / tm2 Updated 2 months ago	0	17	PUBLIC
lfarul / tempmethod2 Updated 2 months ago	0	8	PUBLIC
lfarul / newtempmethod2 Updated 2 months ago	0	6	PUBLIC
lfarul / docker-test	0	0	PUBLIC

On the right side, there are promotional cards for 'Create an Organization', 'Download Docker Desktop', and 'Secure, Private Repo Pricing'. A tip at the bottom left says: 'Tip: Not finding your repository? Try switching namespace via the top left dropdown.'

Widok szczegółowy dla obrazu lfarul/employeeeegistry.



The screenshot shows the detailed view of the Docker image 'lfarul/employeeeegistry'. At the top, there's a header with the repository name, a description (which is empty), and the last pushed time (22 minutes ago). To the right, there's a 'Public View' button and a 'Docker commands' section with the command: `docker push lfarul/employeeeegistry:tagname`. Below the header, there's a 'Tags' section showing a list of tags:

Tag	Icon	Time
18.0		22 minutes ago
17.0		9 hours ago
16.0		a day ago
15.0		2 days ago
14.0		2 days ago

At the bottom of the tags list, there's a link to 'See all'.

Umieszczanie dockerowego obrazu aplikacji na chmurze Google.

```
lfarul@lfarul:~/Pulpit$ docker push gcr.io/nowyprojekt-235718/employeeeeregistry:18.0
The push refers to repository [gcr.io/nowyprojekt-235718/employeeeeregistry]
85c136c86607: Pushed
5ddd7da5e509: Layer already exists
1654ace8426e: Layer already exists
b61c7ac22d6e: Layer already exists
f5d98c1f9470: Layer already exists
270b3f9af07e: Layer already exists
6270adb5794c: Layer already exists
18.0: digest: sha256:432f052b4e67ada49471c4a71614ac0d8786528b35b083842a295bce00c335e7 size: 1784
lfarul@lfarul:~/Pulpit$
```

The screenshot shows the Google Cloud Platform console interface. The top navigation bar includes the Google Cloud Platform logo, the project name 'NowyProjekt', and a search bar. The left sidebar contains navigation options: Storage, Browser, Transfer, Transfer Appliance, and Settings. The main content area is titled 'Object details' and shows the details of a specific object in the 'artifacts.nowyprojekt-235718.appspot.com' bucket. The object is located at 'containers/images/sha256:75fe3778cb4530e479023623ddb0025930c3120caf2b16afd77d8f31ca69278b'. The details include: Access (Not public), Type (application/octet-stream), Size (32.83 MB), Created (June 19, 2019 at 8:23:47 PM UTC+2), Last modified (June 19, 2019 at 8:23:47 PM UTC+2), URI (gs://artifacts.nowyprojekt-235718.appspot.com/containers/images/sha256:75fe3778cb4530e479023623ddb0025930c3120caf2b16afd77d8f31ca69278b), and Link URL (https://storage.cloud.google.com/artifacts.nowyprojekt-235718.appspot.com/containers/images/sha256:75fe3778cb4530e479023623ddb0025930c3120caf2b16afd77d8f31ca69278b).

Podsumowanie i wnioski

Projekt bardzo ciekawy ale i wymagający, integrujący kilka istotnych technologii, takich jak VM, Jenkins, Docker, GitHub, Tomcat oraz bazę danych H2.

Z punktu widzenia poprawności działania projektu należy wyróżnić konfigurację, zarówno po stronie aplikacji – POM.xml, bazy danych – application.properties, jak i po stronie serwera Jenkins – Jenkinsfile oraz Dockera – Dockerfile.

Należy pamiętać, aby porty na jakich pracują serwery nie nakładały się na siebie i zostały odpowiednio wcześniej przekierowane na inne wolne porty.

Bardzo ważne jest także wersjonowanie, szczególnie po stronie obrazu aplikacji jaki się buduje w Dockerfile, jak i po stronie kompilacji. Wersje JDK oraz JRE powinny być spójne po obu stronach, w przeciwnym razie nie uda się uruchomić aplikacji w kontenerze. W tym celu należy sprawdzić w pliku konfiguracyjnym aplikacji – POM.xml jakiej wersji został określony kompilator, w Dockerfile jakiej wersji obraz zostaje zbudowany (FROM) oraz jakiej wersji maven oraz Java jest zainstalowana w naszym środowisku, w tym przypadku na Ubuntu na VM.

Konfiguracja pliku Jenkinsfile w dużym stopniu wynika z tego w jaki sposób przeprowadza się poszczególne stage w terminalu, kiedy nie ma dostępu do serwerów automatyzujących. Np. projekt maven buduje się poprzez `'mvn compile'`, testuje się poleceniem `'mvn test'` a pakuje do pliku .jar poprzez `'mvn package'`. Dotyczy to także budowania oraz uruchamiania dockerowych obrazów – `'docker build -t lfarul/employeeeeregistry:17.0 .'` oraz `'docker push lfarul/employeeeeregistry:17.0'` czy `'docker run -d -p 8282:9000 lfarul/employeeeeregistry:17.0'`.

Aby jednak umieścić dockerowy obraz aplikacji na Docker Hub, Jenkins musi posiadać specjalne credentiale w celu uwierzytelnienia, w przeciwnym razie nie będzie miał dostępu do Docker Hub. Takie credentiale generuje się w GUI Jenkinsa w Pipeline Syntax – Snippet Generator.

Budując obraz dockerowy należy zwrócić uwagę na jakie repozytorium jest przeznaczone. Konwencja nazewnicza jest inna dla Docker Hub oraz Google Cloud Registry.