

Skrypt do warsztatów OCP – aplikacja .NET

Ćwiczenie 1.

Wdrożenie aplikacji .Net opartej na kontenerach Docker

Do pobrania: <https://github.com/chmielu1/kontenery-dockerfile.git>

```
$ mkdir docker
```

```
$ cd docker
```

```
$ git clone https://github.com/chmielu1/dotnetcore.git
```

Budowanie kontenerów

Kontener z bazą danych:

```
$ cd dotnetcore/database_docker  
docker build -t dotnet_database_studentXX .
```

```
$ cat Dockerfile
```

```
From mysql
```

```
LABEL description="This is docker container which deploy database for dotnet application"
```

```
MAINTAINER Mateusz Chmielewski
```

```
ENV MYSQL_ROOT_PASSWORD=qwerty
```

```
COPY ./src/ /docker-entrypoint-initdb.d/
```

Kontener z aplikacją:

```
$ cd ../application_docker/  
$ docker build -t dotnet_application_studentXX .
```

```
$ cat Dockerfile
```

```
FROM microsoft/aspnetcore-build:2.0 AS build-env  
WORKDIR /app
```

```
# Copy csproj and restore as distinct layers
```

```

COPY *.csproj ./
RUN dotnet restore

# Copy everything else and build
COPY . ./
RUN dotnet publish -c Release -o out

# Build runtime image
FROM microsoft/aspnetcore:2.0
WORKDIR /app
COPY --from=build-env /app/out .
ENTRYPOINT ["dotnet", "OPENSIFT.dll"]

```

Wyświetlenie zbudowanych obrazów kontenerów:

```

$ docker images | grep studentXX
dotnet_application_studentXX latest      bd74205b803e    4 minutes ago    312MB
dotnet_database_studentXX latest      4b426bc42a92    11 minutes ago    374MB

```

Import kontenerów do OCP

Stworzenie projektu pod nową aplikację:

```

$ oc login -u studentXX
$ oc new-project dotnet-studentXX

```

Oznaczenie (tag) obrazów tak aby można je było zaimportować do OCP:

```

$ docker tag dotnet_application_studentXX 127.0.0.1:5000/dotnet-
studentXX/dotnet_application:latest

$ docker tag dotnet_database_studentXX 127.0.0.1:5000/dotnet-studentXX/dotnet_database:latest

```

Wyświetlenie żetonu służącego do zalogowania się do wewnętrznego repozytorium OCP

```

$ oc login -u studentXX
$ oc get pods -n default | grep docker-registry
docker-registry-1-27b02 1/1 Running 0 5h

$ oc port-forward docker-registry-1-27b02 -n default 5000:5000 &

$ oc whoami -t
tVVIYCzxr_XcKka6K7ciCEEuLc6fPrylJeDzQlhOfP0

$ docker login -u studentXX -p tVVIYCzxr_XcKka6K7ciCEEuLc6fPrylJeDzQlhOfP0 127.0.0.1:5000
Login Succeeded

```

zaimportowanie obrazów Docker do OCP:

```
$ docker push 127.0.0.1:5000/dotnet-studentXX/dotnet_database:latest
```

```
$ docker push 127.0.0.1:5000/dotnet-studentXX/dotnet_application:latest
```

```
$ pkill ^oc
```

Weryfikacja zapisania obrazów w OCP:

```
$ oc login -u studentXX
```

```
$ oc get images | grep dotnet | grep studentXX
```

```
sha256:6a58648555c6319e60034177b39063f0a6e89c9fe731f5dd81df1f669a7d1726 172.30.1.1:5000/dotnet-s
```

```
tudentXX/dotnet_database@sha256:6a58648555c6319e60034177b39063f0a6e89c9fe731f5dd81df1f669a7d1726
```

```
sha256:f81a637e048951fdef4d6960b96c69059311731a9f148575812141720c856ee8 172.30.1.1:5000/dotnet-s
```

```
tudentXX/dotnet_application@sha256:f81a637e048951fdef4d6960b96c69059311731a9f148575812141720c856ee
```

Wdrożenie aplikacji

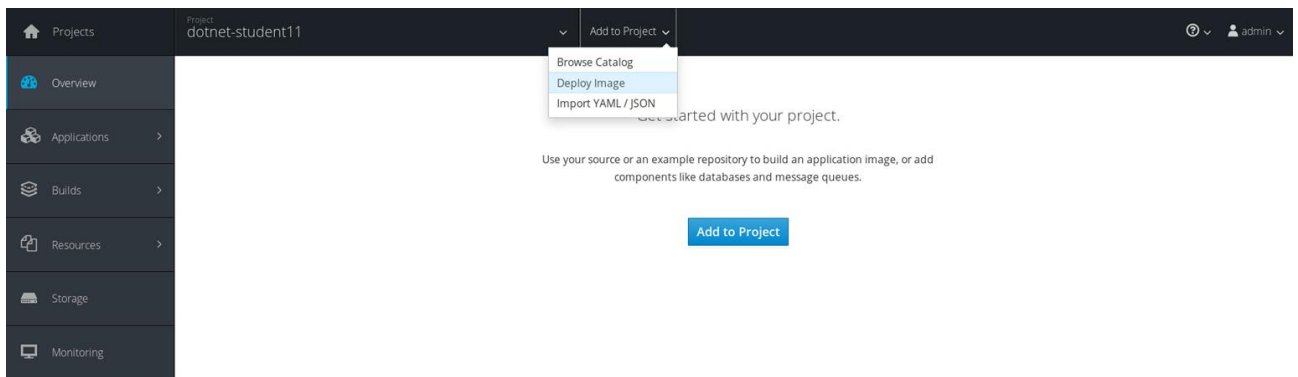
Uruchomienie konsoli OCP

<https://master1.lab19.example.com:8443>

UWAGA! Należy zaakceptować certyfikat typu self-signed konsoli OCP.

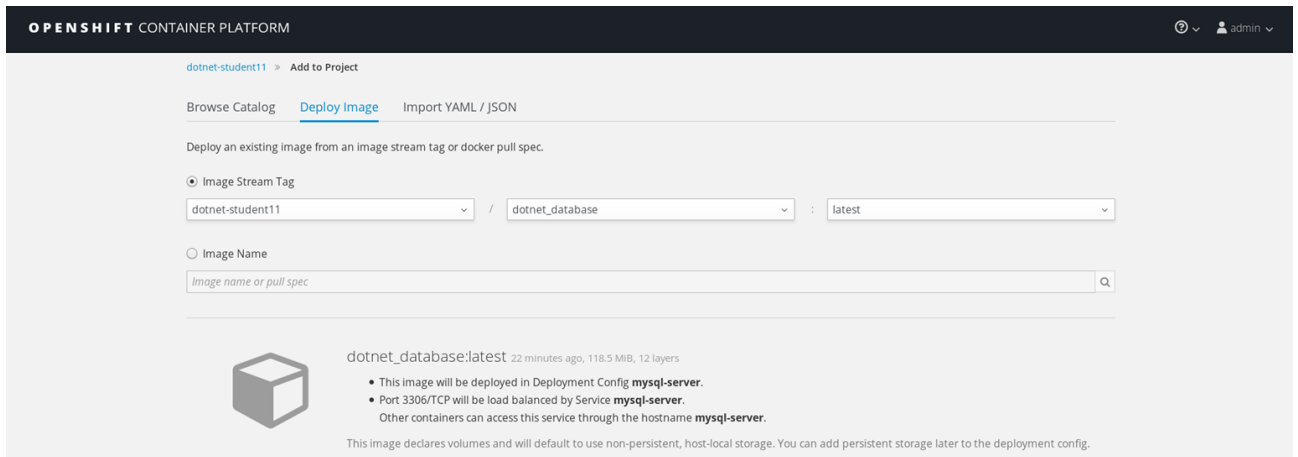
Logujemy się jako studentXX używając hasła podanego przez instruktora i wchodzimy w kontekst projektu dotnet-studentXX

Wybieramy z menu „Add to project” opcję „Deploy image”

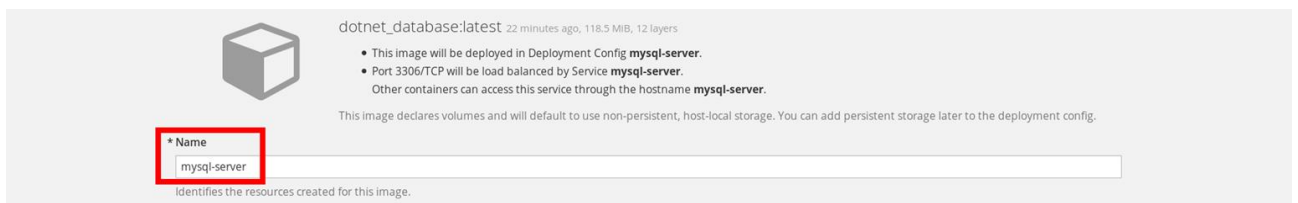


Wybieramy strumień obrazów (image stream tag):

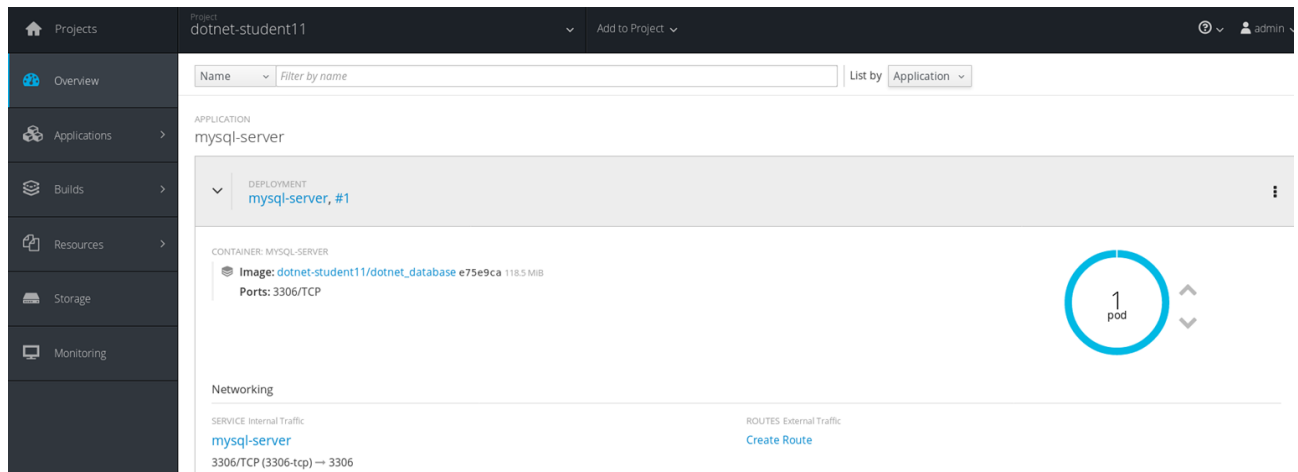
dotnet-studentXX/dotnet_database:latest



Następnie **należy koniecznie** wypełnić pole **Name** wartością „mysql-server”, dzięki temu pody będą mogły się ze sobą komunikować używając wskazanej nazwy.



Po kilku chwilach pod powinien być w pełni funkcjonalny.



Po wdrożeniu bazy danych rozszerzamy uprawnienia dla kontenera aplikacji

```
$ oc login -u studentXX
```

```
$ oc adm policy add-scc-to-user anyuid -z default -n dotnet-studentXX
```

Następnie wdrażamy aplikację - ponownie z menu „Add to project”, należy wybrać opcję „Deploy image” (prawy górny róg ilustracji).

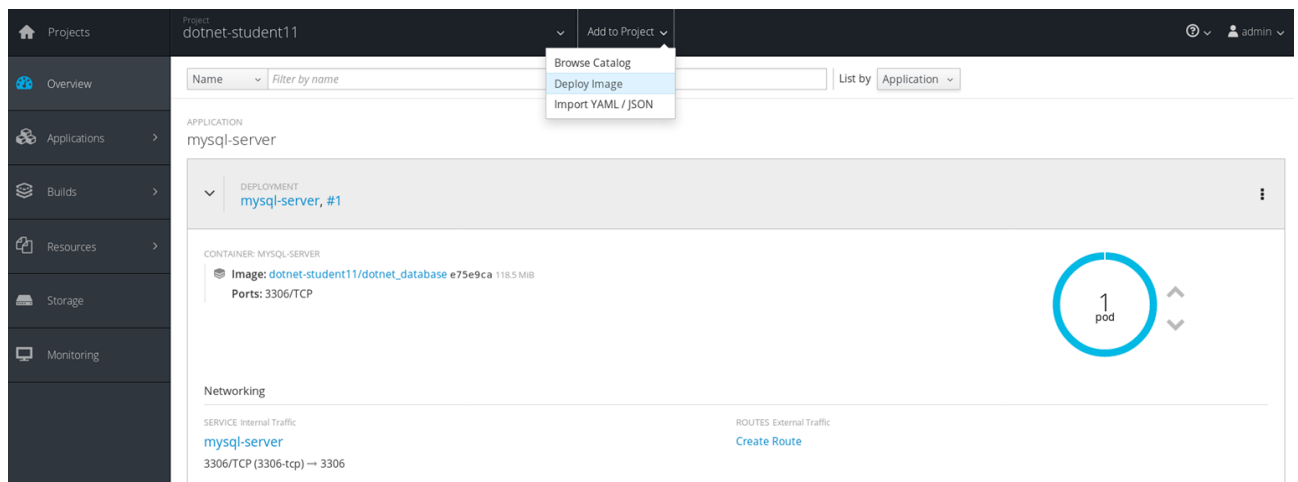


Image stream tag należy wybrać tak, jak na poniższej ilustracji, uwzględniając zróżnicowany parametr studentXX

OPENSIFT CONTAINER PLATFORM admin

[dotnet-student11](#) » [Add to Project](#)


[Browse Catalog](#) [Deploy Image](#) [Import YAML / JSON](#)

Deploy an existing image from an image stream tag or docker pull spec.

☒ Image Stream Tag

/ :

☐ Image Name




dotnet_application:latest 24 minutes ago, 128.3 MiB, 7 layers

- This image will be deployed in Deployment Config **<name>**.
- Port 80/TCP will be load balanced by Service **<name>**.

Other containers can access this service through the hostname **<name>**.

Pole „Name” należy wypełnić jak poniżej > dotnet-application



dotnet_application:latest 25 minutes ago, 128.3 MiB, 7 layers

- This image will be deployed in Deployment Config **dotnet-application**.
- Port 80/TCP will be load balanced by Service **dotnet-application**.

Other containers can access this service through the hostname **dotnet-application**.

*** Name**

Identifies the resources created for this image.

Pull Secret

Kolejnym krokiem będzie umożliwienie podłączenia się do aplikacji z zewnątrz (ekspozycja punktu dostępu do aplikacji), poprzez stworzenie „ścieżki” (route) do aplikacji.

The screenshot shows the OpenShift console interface. On the left is a sidebar with navigation links: Projects, Overview, Applications, Builds, Resources, Storage, and Monitoring. The main area displays the details of a deployment named 'dotnet-application, #1'. It shows the container image 'dotnet-student11/dotnet_application a4f9e76' and the ports '80/TCP'. A red box highlights the 'Create Route' button under the 'ROUTES External Traffic' section.

Wszystkie wartości należy pozostawić domyślnie.

Aplikacja już powinna być dostępna pod nowoutworzonym adresem.

<http://dotnet-application-dotnet-student11.apps.lab19.example.com>

The screenshot shows the OpenShift web application interface. The header has 'OPENSIFT' and links to 'Home', 'About', and 'Contact'. The main content area has the title 'DOTNETCORE for OPENSIFT' and the text 'OPENSIFT with us' and 'Docker with us'. Below this is a section titled 'Role' with a table listing roles and their names.

Role	Name
ROLE_ADMIN	Administrator
ROLE_CONSULTANT	Consultant
ROLE_MANAGER	Manager
ROLE_PROJECTMANAGER	PM
ROLE_REPORT	Report role

© 2018 - LinuxPolska

Ćwiczenie 2.

Ciągłe dostarczanie (CD) dla kontenerów. Aplikacja .NET

Bazując na poprzednim przykładzie.

Pobieramy gałąź (branch) „test” ze zmodyfikowaną wersją kodu źródłowego:

```
$ cd
$ cd docker
$ mkdir test
$ cd test
$ git clone -b test https://github.com/chmielu1/dotnetcore.git
```

Następnie przechodzimy do Dockerfile’a aplikacji i budujemy nowy obraz kontenera aplikacji:

```
$ cd dotnetcore/application_docker
$ docker build -t dotnet_application_studentXX .
```

Wyświetlenie zbudowanego obrazu

```
$ docker images | grep dotnet_application_studentXX
dotnet_application_student11   latest      c8f6f4d7b277    23 seconds ago 312MB
```

Oznaczenie (tag) zbudowanego obrazu:

```
$ docker tag dotnet_application_studentXX:latest 127.0.0.1:5000/dotnet-
studentXX/dotnet_application:latest

$ oc port-forward docker-registry-1-27b02 -n default 5000:5000 &

$ oc whoami -t
tVVIYCzxr_XcKka6K7ciCEEuLc6fPrylJeDzQlhOfP0

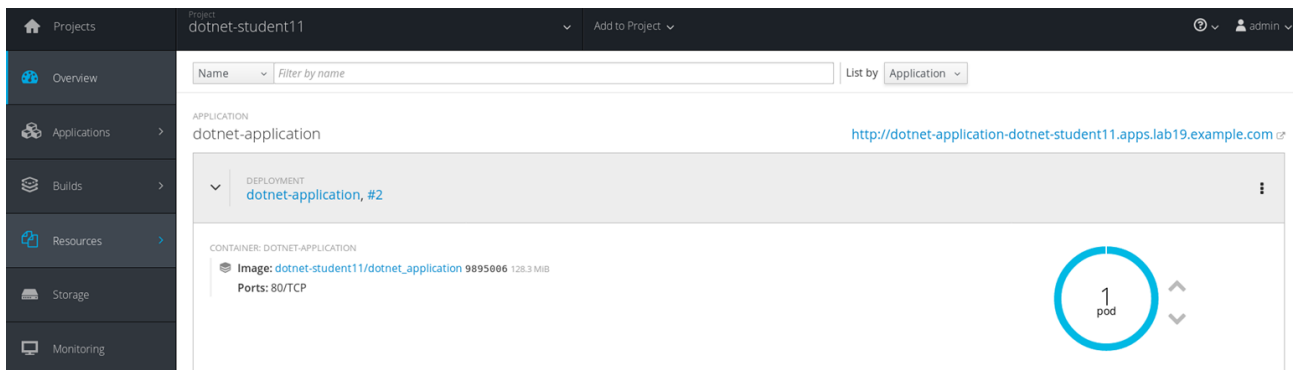
$ docker login -u studentXX -p tVVIYCzxr_XcKka6K7ciCEEuLc6fPrylJeDzQlhOfP0 127.0.0.1:5000
Login Succeeded

$ docker push 127.0.0.1:5000/dotnet-studentXX/dotnet_application:latest

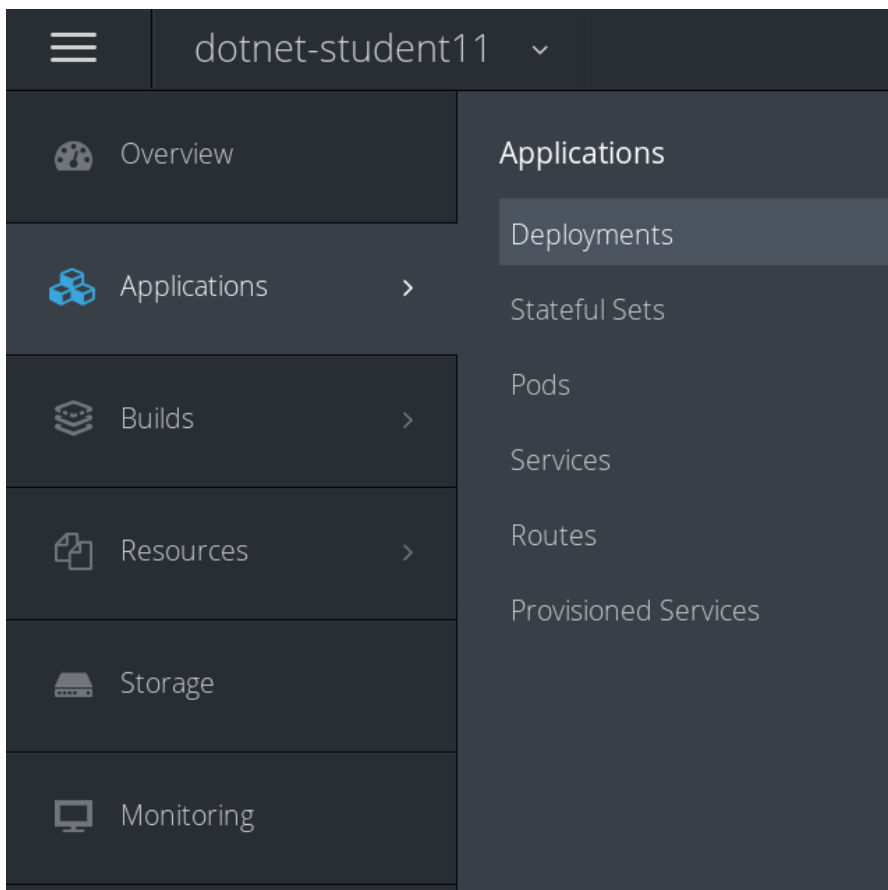
$ pkill ^oc
```


OCP zauważając nową wersję obrazu kontenera w swoim rejestrze wykona automatyczne podniesienie środowiska wykorzystując nowy kontener:

Poniżej slajd z nową wersją poda



Następnie wchodzimy w zakładkę Applications > Deployments



Poniżej wpis nowego wdrożenia:

Deployment	Status	Created	Trigger
#2 (latest)	Active, 1 replica	a minute ago	Image change
#1	Complete	8 minutes ago	Image change

Następnie otwieramy okno prywatne przeglądarki lub czyścimy pamięć podręczną przeglądarki i przechodzimy do adresu URL aplikacji (poniżej).

<http://dotnet-application-dotnet-student11.apps.lab19.example.com>

Nowa wersja aplikacji powinna wyglądać jak na poniższej ilustracji.

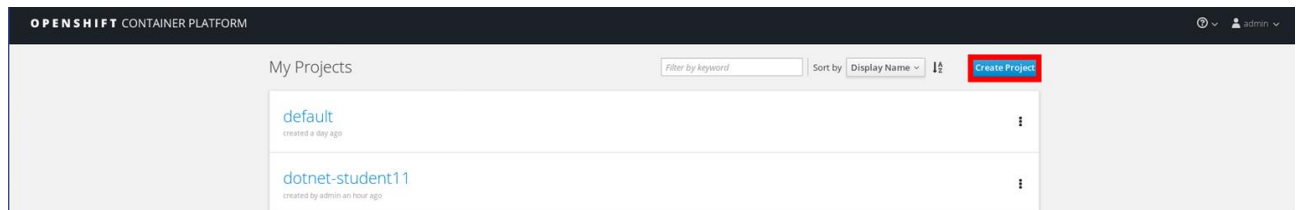
Role	Name
ROLE_ADMIN	Administrator
ROLE_CONSULTANT	Consultant
ROLE_MANAGER	Manager
ROLE_PROJECTMANAGER	PM
ROLE_REPORT	Report role

© 2018 - LinuxPolska -> nowawersja produkcyjna

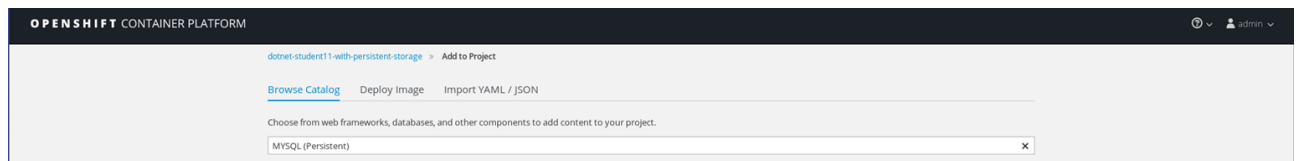
Ćwiczenie 3.

Wykorzystanie wzorca bazy danych.

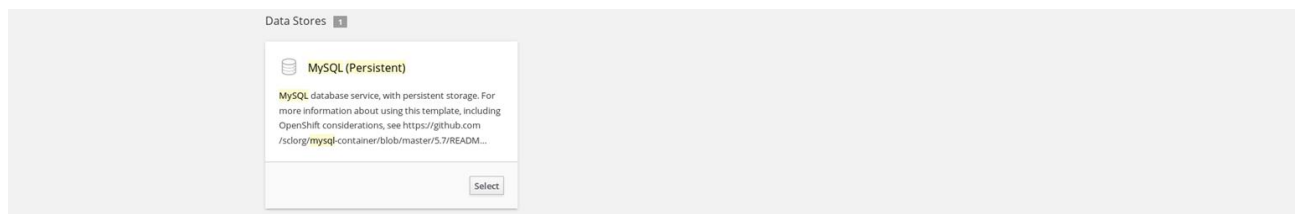
Tworzymy nowy projekt dotnet-studentXX-with-persistent-storage i przechodzimy do jego kontekstu.



Następnie wyszukiujemy wzorec bazy danych Mysql z nieulotną pamięcią




(...)



Na poniższych ilustracjach przedstawiono prawidłowe wypełnienie odpowiednich wartości.

OPENSIFT CONTAINER PLATFORM




MySQL (Persistent)

MySQL database service, with persistent storage. For more information about using this template, including OpenShift considerations, see <https://github.com/sciorg/mysql-container/blob/master/5.7/README.md>.

NOTE: Scaling to more than one replica is not supported. You must have persistent volumes available in your cluster to use this template.

Images

 mysql5.7 from parameter Version of MySQL Image

Parameters

*** Memory Limit**

Maximum amount of memory the container can use.

Namespace

The OpenShift Namespace where the ImageStream resides.

*** Database Service Name**

The name of the OpenShift Service exposed for the database.

*** MySQL Connection Username**

*** MySQL Connection Password**

Password for the MySQL connection user.

*** MySQL root user Password**

Password for the MySQL root user.

*** MySQL Database Name**

Name of the MySQL database accessed.

*** Volume Capacity**

Volume space available for data, e.g. 512M, 2Gi.

*** Version of MySQL Image**

Version of MySQL Image to be used (5.5, 5.6, 5.7, or latest).

Labels

[About Labels](#)

The following labels are being added automatically. If you want to override them, you can do so below.

template	mysql-persistent-template
app	mysql-persistent

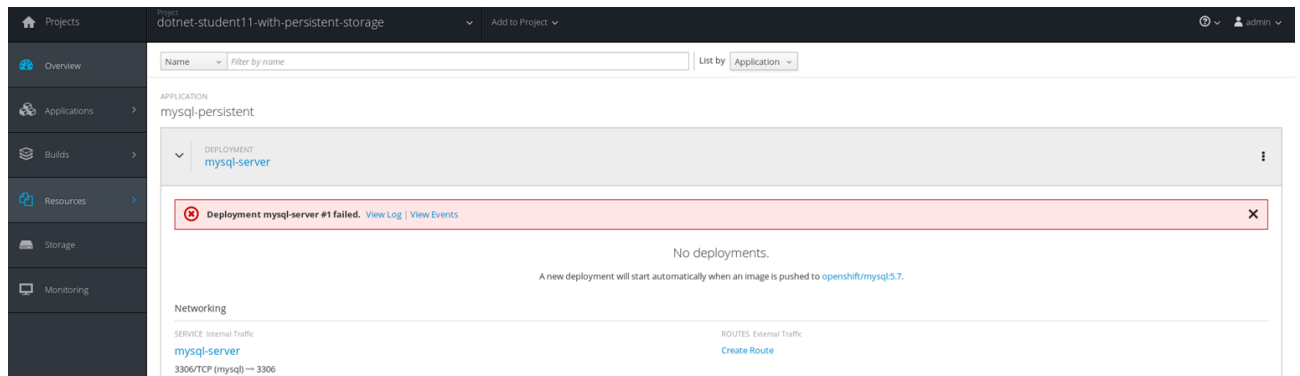
Each label is applied to each created resource.

Name	Value

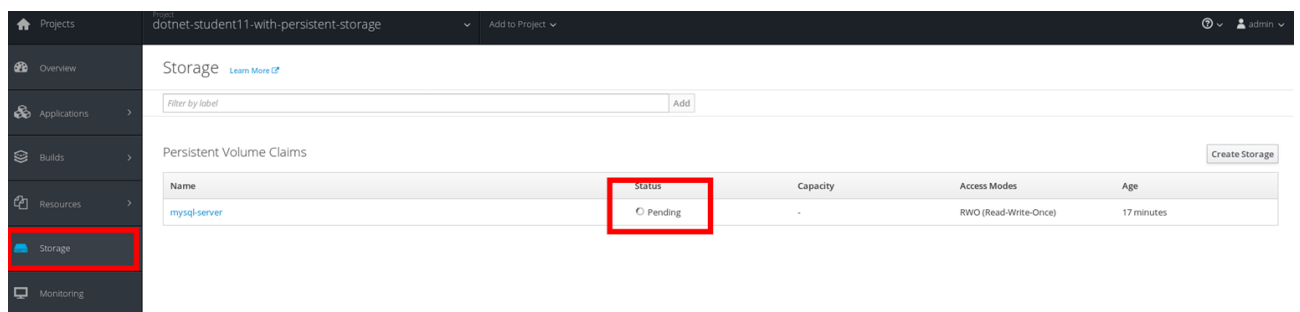
[Add Label](#)

[Create](#) [Cancel](#)

Z powodu niezadeklarowania dla poda pamięci stałej, kontener z bazą mysql się nie wybuduje.



Stan wdrażania pamięci stałym możemy zobaczyć w zakładce Storage



Aby naprawić problem należy stworzyć zasób nfs, który będzie dostępny dla wskazanego poda.
(pamiętając o parametryzacji dla każdego uczestnika warsztatu)

w pliku `/etc/exports`

```
/var/export/vol-studXX *(rw,async,all_squash)
```

```
$ mkdir -p /var/export/vol-studXX
```

```
$ chmod 700 /var/export/vol -studXX
```

```
$ chown nfsnobody:nfsnobody /var/export/vol-studXX
```

```
$ chcon -R -t svirt_sandbox_file_t /var/export/vol-studXX
```

```
$setsebool -P virt_use_nfs=true
```

```
$setsebool -P virt_sandbox_use_nfs=true
```

```
$ exportfs -a
```

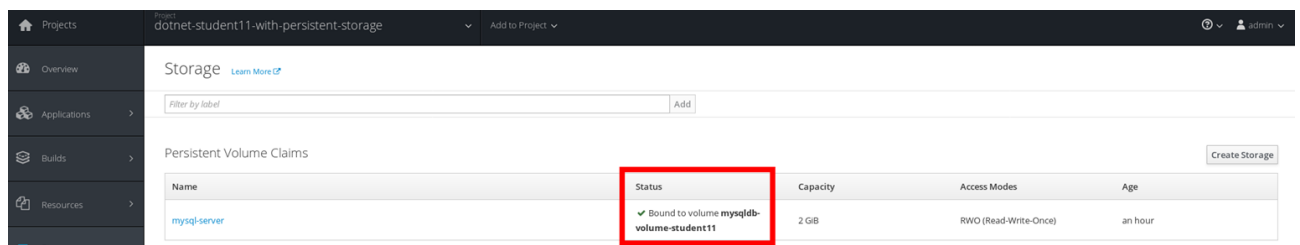
```
$ showmount -e
```

```
$ cd
$ vi mysqlldb-volume.json

{
  "apiVersion": "v1",
  "kind": "PersistentVolume",
  "metadata": {
    "name": "mysqlldb-volume-studentXX",
    "labels": {
      "name": "mysqlldb-volume-studentXX"
    }
  },
  "spec": {
    "capacity": {
      "storage": "2Gi"
    },
    "accessModes": [ "ReadWriteOnce" ],
    "nfs": {
      "path": "/var/export/vol-studXX",
      "server": "admin1.lab19.example.com"
    }
  }
}
```

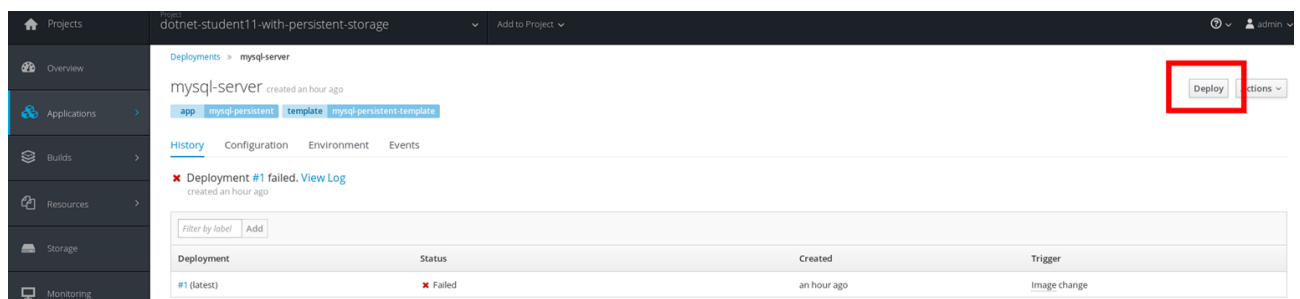
```
$ oc create -f mysqlldb-volume.json
```

Po stworzeniu zasobu Persistent Volume, OCP automatycznie spróbuje podpiąć zasób do projektu.



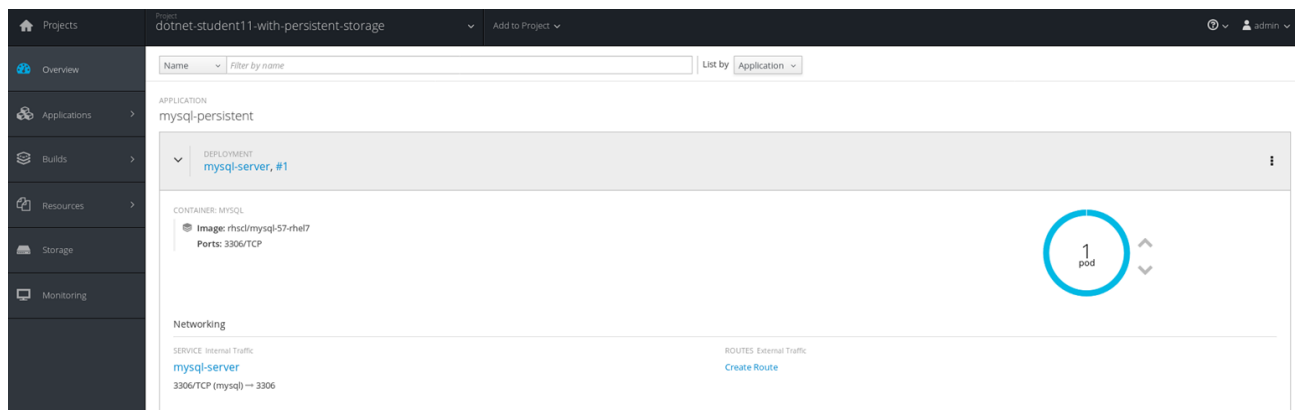
Name	Status	Capacity	Access Modes	Age
mysql-server	✓ Bound to volume mysqlldb-volume-student11	2 GiB	RWO (Read-Write-Once)	an hour

Należy ponownie wymusić deployment poda. Poprzez zakładkę Applications -> Deployments -> mysql-server następnie klikając Deploy



Deployment	Status	Created	Trigger
#1 (latest)	✗ Failed	an hour ago	Image change

Poniższy rysunek przedstawia nowoutworzoną instancję bazy danych



Pliki skryptów SQL dostępne są pod adresem:

<https://github.com/chmielu1/sql.git>

Logujemy się po ssh i ściągamy repozytorium:

```
$ cd
$ git clone https://github.com/chmielu1/sql.git
$ cd sql
```

Instalujemy klienta mysql a następnie wykonujemy skrypty SQL:

```
$ yum -y install mariadb

$ oc project dotnet-student11-with-persistent-storage

$ oc get pods
NAME                READY   STATUS    RESTARTS   AGE
mysql-server-1-6dk7b 1/1     Running   0           56m

$ oc port-forward mysql-server-1-6dk7b 3306:3306 &
$ mysql -h 127.0.0.1 -u ehour -pqwerty ehour < fresh.mysql.sql

$ pkill ^oc
```

Następnie wdrażamy poda aplikacji dotnet

```
$ docker tag dotnet_application_studentXX:latest 127.0.0.1:5000/dotnet-studentXX-
with-persistent-storage/dotnet_application:latest

$ oc port-forward docker-registry-1-27b02 -n default 5000:5000 &

$ oc whoami -t
tVVIYCzxr_XcKKa6K7ciCEEuLc6fPrylJeDzQlhOfP0
```

```
$ docker login -u studentXX -p tVVIYCzxr_XcKka6K7ciCEEuLc6fPrylJeDzQlhOfP0 127.0.0.1:5000
Login Succeeded
```

```
$ docker push 127.0.0.1:5000/dotnet-student11-with-persistent-storage/dotnet_application:latest
```

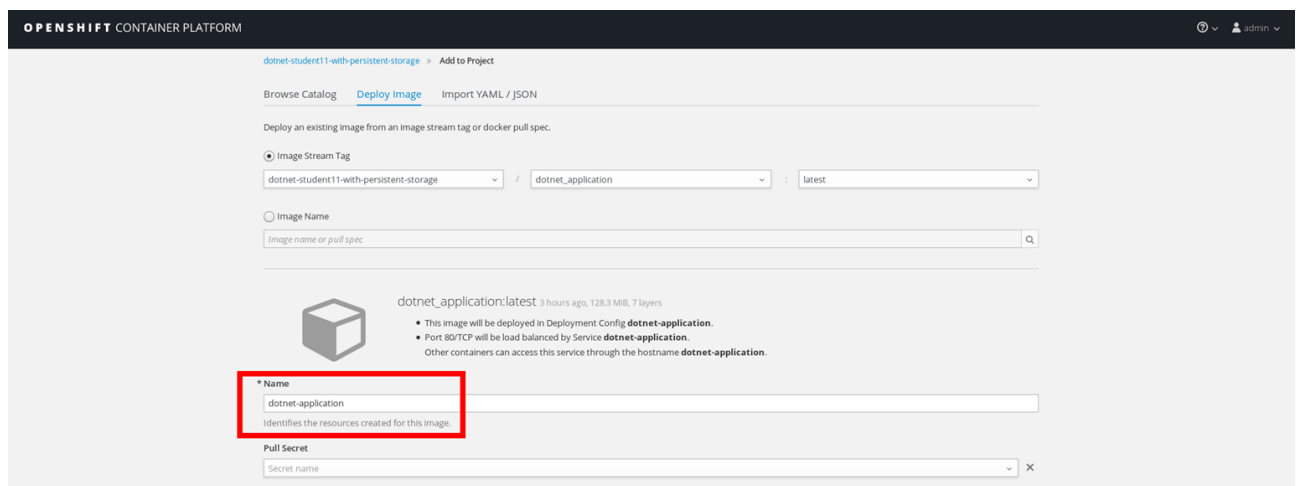
Następnie rozszerzamy uprawnienia dla kontenera aplikacji

```
$ oc login -u studentXX
```

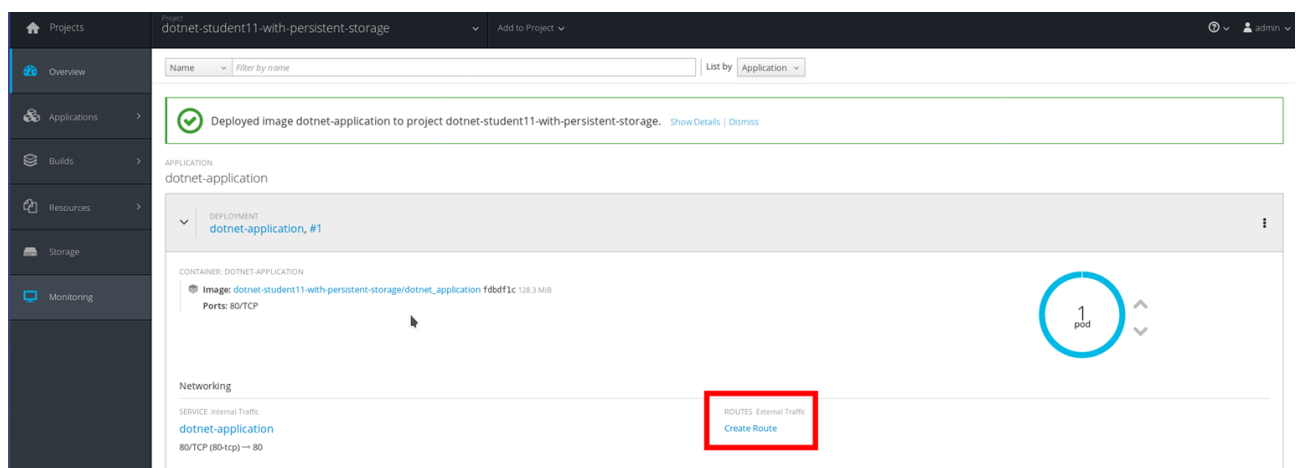
```
$ oc adm policy add-scc-to-user anyuid -z default -n dotnet-studentXX-with-persistent-storage
```

Przechodzimy z powrotem do web panelu i startujemy pod aplikacją dotnet.

Wybieramy „Add to Project”, a następnie „Deploy Image”.



Na koniec dodajemy ścieżkę (pozostawiamy wartości domyślne) do aplikacji i weryfikujemy poprawność działania aplikacji:



<http://dotnet-application-dotnet-student11-with-persistent-storage.apps.lab19.example.com/>

OPENSIFT Home About Contact

DOTNETCORE for OPENSIFT

OPENSIFT with us

Docker with us

Role

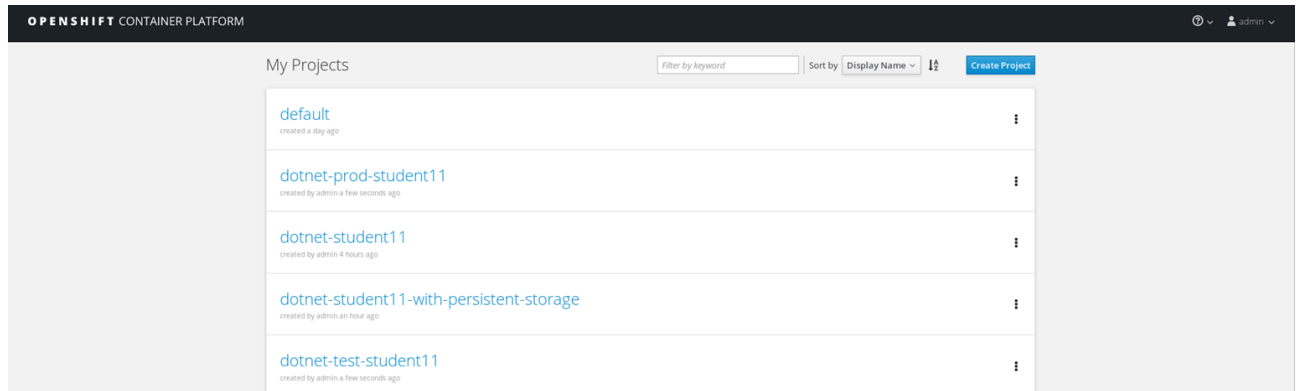
Role	Name
ROLE_ADMIN	Administrator
ROLE_CONSULTANT	Consultant
ROLE_MANAGER	Manager
ROLE_PROJECTMANAGER	PM
ROLE_REPORT	Report role

© 2018 - LinuxPolska -> nowawersja produkcyjna

Ćwiczenie 4.

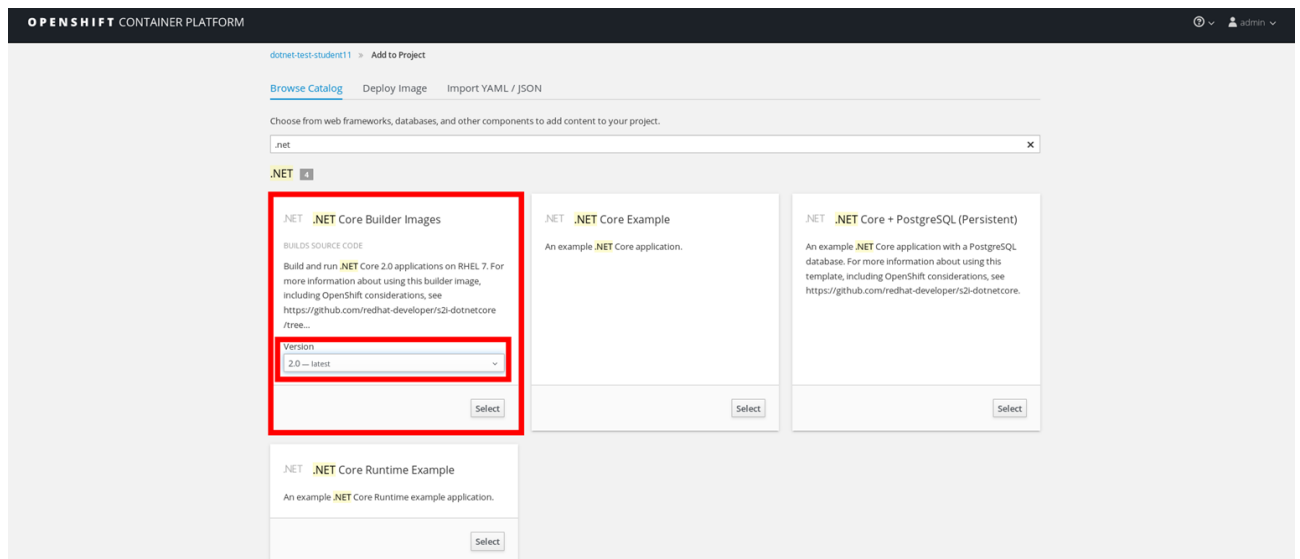
Wykorzystanie mechanizmu source to image dla aplikacji .NET

Tworzymy dwa nowe projekty dotnet-test-studentXX oraz dotnet-prod-studentXX odpowiednio dla środowiska testowego oraz produkcyjnego.



Następnie korzystając z wzorca s2i (source-to-image) budujemy środowisko testowe.

Klikamy „Add To Project” następnie wyszukujemy wzorzec .net



Klikamy "select" a następnie wybieramy opcje „advanced options”

OPENSIFT CONTAINER PLATFORM

dotnet-test-student11 > Add to Project > Catalog > .NET Core Builder Images

.NET

.NET Core Builder Images

Build and run .NET Core 2.0 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/redhat-developer/s2i-dotnetcore/tree/master/2.0/build/README.md>.

Version: 2.0

* Name

 Identifies the resources created for this application.

* Git Repository URL

 Sample repository for dotnet: <https://github.com/redhat-developer/s2i-dotnetcore-ex.git>, ref: dotnetcore-2.0, context dir: app
[Try it ↗](#)

Show **advanced options** for source, routes, builds, and deployments.

Create Cancel

Następnie uzupełniamy pole „Name” np. „dotnet”, wpisujemy URLa repozytorium Gita oraz wybieramy odpowiedniego gałąź (branch), „Context Dir” reszta wartości pozostawiamy domyślne.

Git Repository URL:

<https://github.com/chmielu1/dotnetcore.git>

Git Reference:

nodb-test

Context Dir:

application_docker

OPENSIFT CONTAINER PLATFORM

dotnet-test-student11 > Add to Project > Catalog > .NET Core Builder Images

.NET

.NET Core Builder Images

Build and run .NET Core 2.0 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/redhat-developer/s2i-dotnetcore/tree/master/2.0/build/README.md>.

Version: 2.0

* Name

 Identifies the resources created for this application.

* Git Repository URL

 Sample repository for dotnet: <https://github.com/redhat-developer/s2i-dotnetcore-ex.git>, ref: dotnetcore-2.0, context dir: app
[Try it ↗](#)

Git Reference

 Optional branch, tag, or commit.

Context Dir

 Optional subdirectory for the application source code, used as the context directory for the build.

Source Secret
 X
 Secret with credentials for pulling your source code. [Learn More ↗](#)
[Create New Secret](#)

Routing
☒ Create a route to the application [About Routing](#)

Na koniec klikamy na samym dole strony pole „create”.

Po chwili powinniśmy mieć działającego poda, co możemy sprawdzić wchodząc na urla

<http://dotnet-dotnet-prod-student11.apps.lab19.example.com/>

OPENSIFT Home About Contact

DOTNETCORE for OPENSIFT

OPENSIFT with us

Docker with us

OPENSIFT APP - TESTING ENVIRONMENT

© 2018 - LinuxPolska

W analogiczny sposób tworzymy środowisko produkcyjne, pamiętając żeby przejść na projekt środowiska produkcyjnego dotnet-prod-studentXX oraz uzupełnienie konfiguracji wzorca w analogiczny sposób:

Name:

dotnet

Git Repository URL:

<https://github.com/chmielu1/dotnetcore.git>

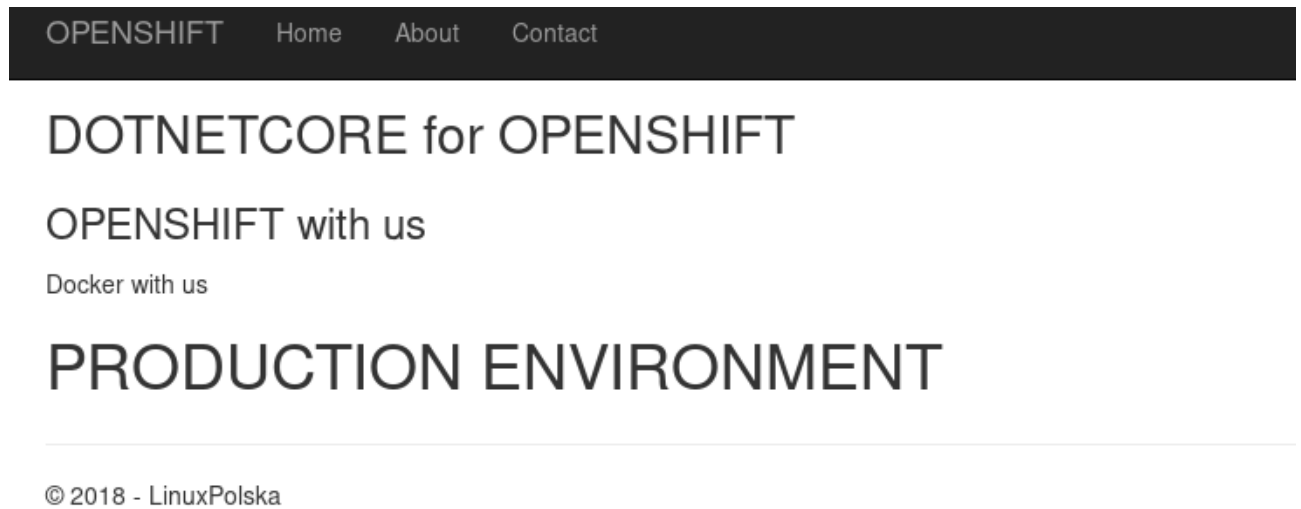
Git Reference:

nodb-prod

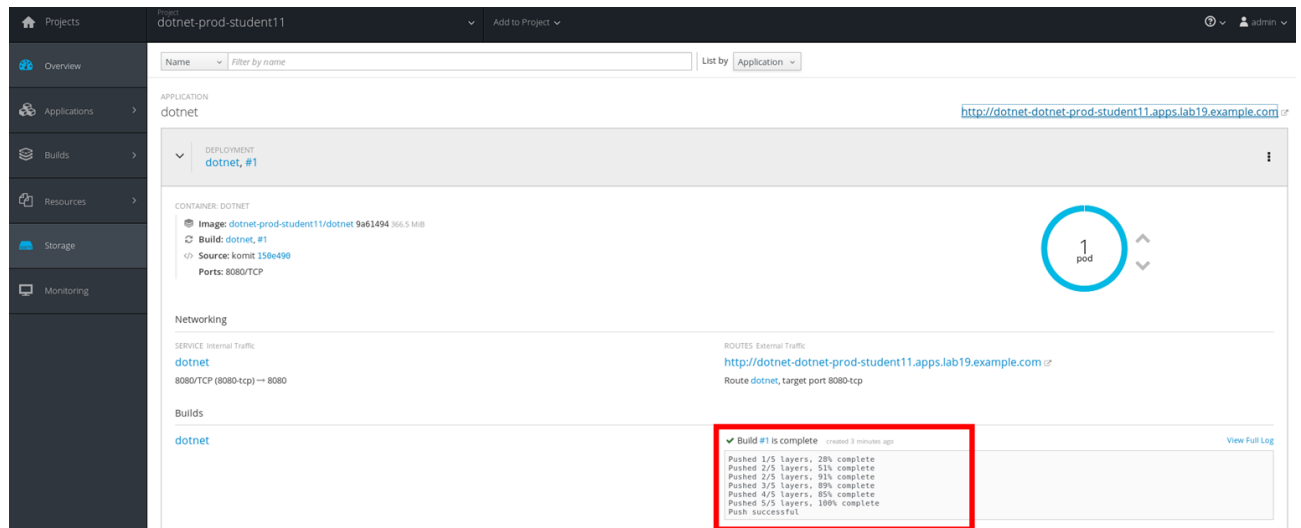
Context Dir:

application_docker

Po wejściu na url <http://dotnet-dotnet-test-student11.apps.lab19.example.com/> powinniśmy uzyskać następujący widok:



Poniższa ilustracja przedstawia proces budowania się obrazu metodą s2i, który składa się ze ściągnięcia odpowiedniego obrazu kontenera następnie zbudowania w nim aplikacji.



Ćwiczenie 5.

Wykonanie zmiany w aplikacji przy użyciu Git.

W tym ćwiczeniu wykorzystujemy oba poprzednie projekty testowy i produkcyjny.

Najpierw pobieramy kod ze środowiska testowego z repozytorium Git.

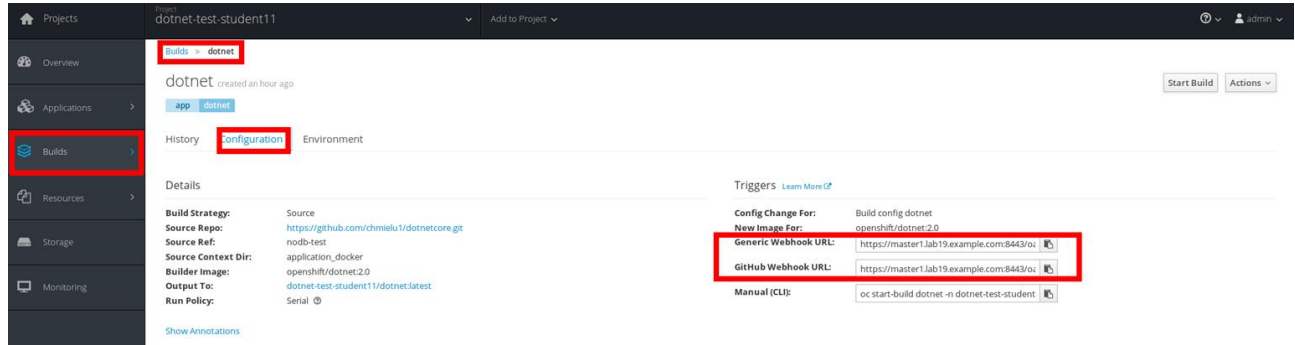
```
$ cd
$ mkdir dotnet
$ cd dotnet
$ git clone https://github.com/chmielu1/dotnetcore.git -b nodb-test
$ cd dotnetcore/application_docker
$ sed -i 's/LinuxPolska/LinuxPolska -> nowawersja produkcyjna/' Views/Shared/_Layout.cshtml
$ git add .

$ git commit -m "openshift"

$ git push
```

Następnie przechodzimy na projekt testowy dotnet-test-studentXX!

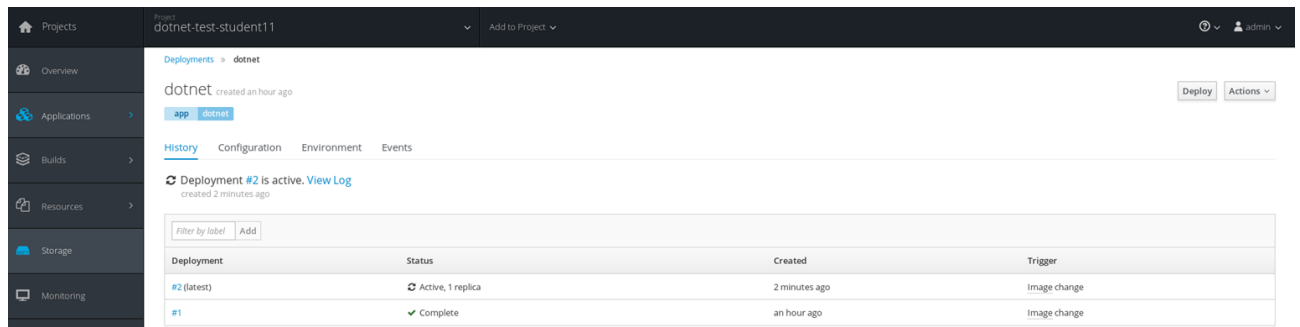
Aby uruchomić ponowne budowanie obrazu zawierającego nowy kod aplikacji należy skorzystać z webhooka, który znajduje się w menu Builds -> dotnet-> Configuration



Następnie ręcznie uruchamianie generic webhooka:

```
$ curl -X POST -k https://master1.lab19.example.com:8443/oapi/v1/
amespaces/dotnet-test-student11/buildconfigs/dotnet/webhooks/e163e15f68e6db61/generic
```

W zakładce Applications -> Deployments możemy zauważyć nowe wdrożenie.



Możemy teraz obserwować jak uruchamia się nowy pod

```
$ oc get pods -n dotnet-test-studentXX
NAME          READY   STATUS    RESTARTS   AGE
dotnet-1-build 0/1     Error     0          1h
dotnet-2-6zlfz 1/1     Running   0          2m
dotnet-2-build 0/1     Completed 0          59m
dotnet-3-build 0/1     Completed 0          3m
```

Następnie można sprawdzić poprawność wprowadzonej zmiany wchodząc przez przeglądarkę na podany poniżej URL, lub programem curl:

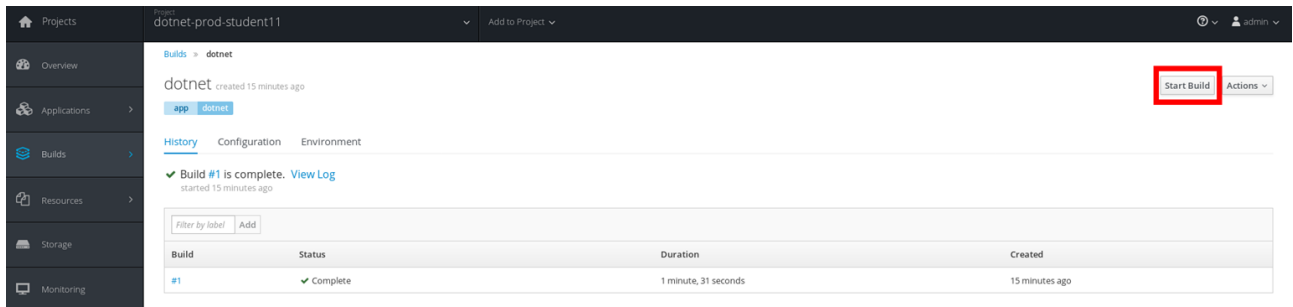
<http://dotnet-dotnet-test-student11.apps.lab19.example.com/>

```
$ curl http://dotnet-dotnet-test-student11.apps.lab19.example.com/ 2> /dev/
null | grep Linux
<p>&copy; 2018 - LinuxPolska -> nowawersja produkcyjna</p>
```

Dostarczamy zawartość gałęzi testowej na środowisko produkcyjne

```
$ cd
$ cd dotnet/dotnetcore
$ git checkout -t origin/nodb-prod
$ git merge nodb-test -X theirs -m "wprowadzanie zmian na produkcji"
$ git add .
$ git commit -m „openshift”
$ git push
```

Ręcznie uruchamiamy propagację zmian na środowisko produkcyjne poprzez Builds -> Builds -> dotnet-> Start Build:

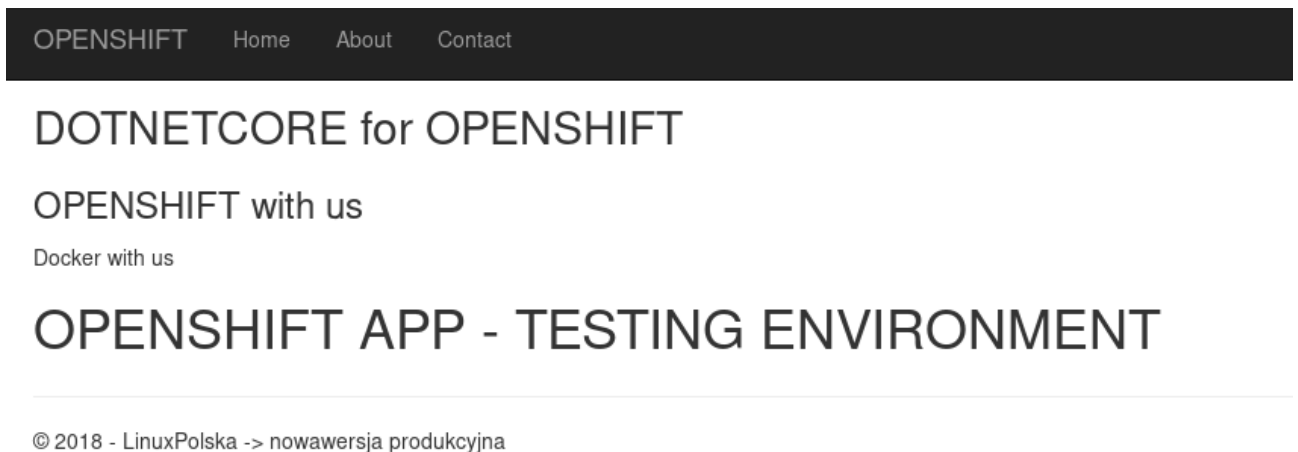


The screenshot shows the OpenShift console interface. On the left is a sidebar with navigation links: Projects, Overview, Applications, Builds, Resources, Storage, and Monitoring. The main area displays the 'dotnet' application page. At the top right, there is a 'Start Build' button highlighted with a red rectangle, next to an 'Actions' dropdown menu. Below this, a message states 'Build #1 is complete. View Log'. A table below shows the build details:

Build	Status	Duration	Created
#1	Complete	1 minute, 31 seconds	15 minutes ago

Po zbudowaniu poda sprawdzamy zbudowane zmiany np. poprzez przeglądarkę.

<http://dotnet-dotnet-prod-student11.apps.lab19.example.com/>



The screenshot shows the OpenShift website. The header is dark with links: OPENSIFT, Home, About, and Contact. The main content area has the following text:

DOTNETCORE for OPENSIFT

OPENSIFT with us

Docker with us

OPENSIFT APP - TESTING ENVIRONMENT

© 2018 - LinuxPolska -> nowawersja produkcyjna

Ćwiczenie 6.

Skalowanie aplikacji .Net

Aplikację produkcyjną możemy wyskalować na dwa sposoby

CLI

Wykonujemy następującą sekwencję poleceń

```
$ oc login -u studentXX
$ oc project dotnet-prod-studentXX
$ oc scale --replicas=5 dc dotnet
deploymentconfig "php-prod" scaled
```

```
$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
dotnet-1-build	0/1	Completed	0	31m
dotnet-2-65jvz	1/1	Running	0	12s
dotnet-2-build	0/1	Completed	0	2m
dotnet-2-m2rb5	1/1	Running	0	12s
dotnet-2-r7nd9	1/1	Running	0	12s
dotnet-2-r87hs	1/1	Running	0	2m
dotnet-2-tf755	1/1	Running	0	12s

Konsola OCP

Drugim sposobem skalowania jest użycie konsoli OCP. Zaznaczone poniżej strzałki umożliwiają skalowanie w górę lub w dół do zadanej ilości podów.

The screenshot shows the OpenShift Console interface for the project 'dotnet-prod-student11'. The left sidebar contains navigation links: Projects, Overview, Applications, Builds, Resources, Storage, and Monitoring. The main content area shows the application 'dotnet' with a deployment 'dotnet, #2'. The deployment details include the container image 'dotnet-prod-student11/dotnet 10d220a 166.5 MiB', build 'dotnet, #2', and source 'wprowadzanie zmian na produkcji 72e88e3'. The deployment status is 'Running' with 1 pod. A red box highlights the up and down arrows next to the pod count, indicating the scaling controls.

The screenshot shows the OpenShift Console interface for the project 'dotnet-prod-student11'. The left sidebar contains navigation links: Projects, Overview, Applications, Builds, Resources, Storage, and Monitoring. The main content area shows the application 'dotnet' with a deployment 'dotnet, #2'. The deployment details include the container image 'dotnet-prod-student11/dotnet 10d220a 166.5 MiB', build 'dotnet, #2', and source 'wprowadzanie zmian na produkcji 72e88e3'. The deployment status is 'Running' with 5 pods. A red box highlights the up and down arrows next to the pod count, indicating the scaling controls.

Poniżej lista podów:

Name	Status	Containers Ready	Container Restarts	Age
dotnet-2-50xn9	Running	1/1	0	a minute
dotnet-2-7s5wc	Running	1/1	0	a minute
dotnet-2-b24kk	Running	1/1	0	a minute
dotnet-2-mng5	Running	1/1	0	a minute
dotnet-2-fb6d5	Running	1/1	0	3 minutes
dotnet-2-build	Completed	0/1	0	5 minutes
dotnet-1-build	Completed	0/1	0	20 minutes

Ćwiczenie 7.

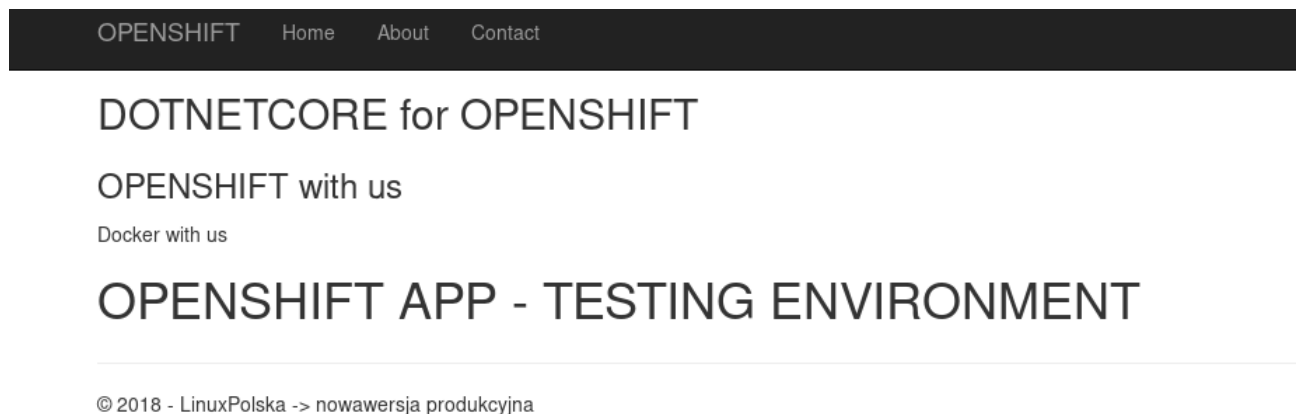
Wycofywanie zmian aplikacji .Net

Ponownie należy skorzystać z aplikacji produkcyjnej wygenerowanej poprzednio.

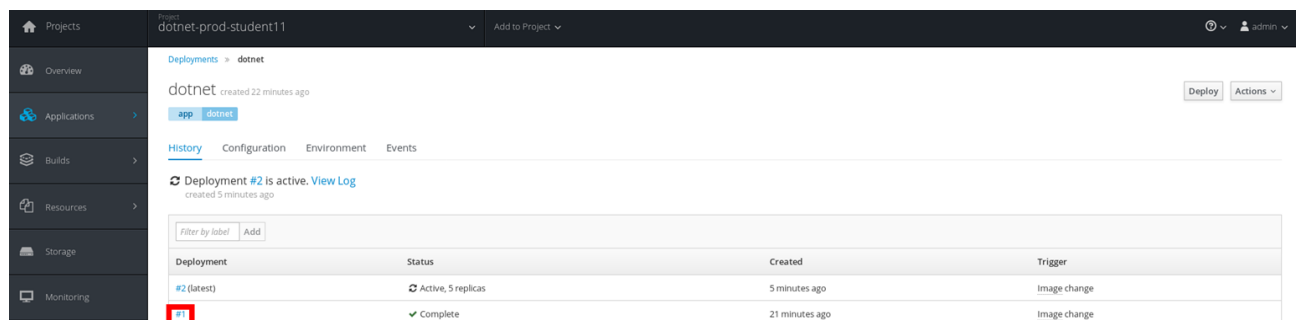
Na początek sprawdzamy obecną zawartość strony:

```
$ curl http://dotnet-dotnet-prod-student11.apps.lab19.example.com/ 2> /dev/null | grep "Linux\|ENVIRONMENT"
```

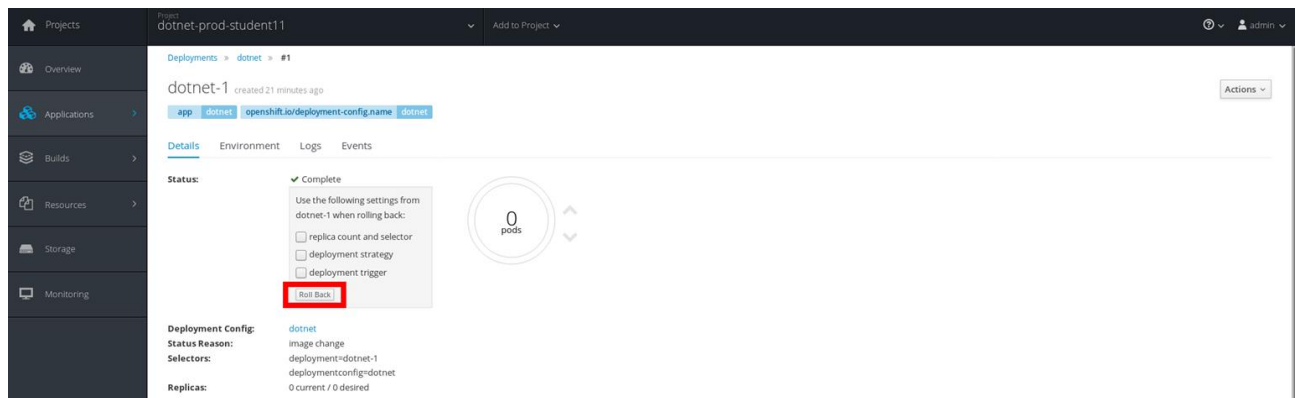
<h1> OPENSIFT APP - TESTING ENVIRONMENT </h1>
<p>© 2018 - LinuxPolska -> nowawersja produkcyjna</p>



Następnie przechodzimy w projekcie do zakładki Applications -> Deployments -> dotnet i klikamy wdrożenie #1



Aby wykonać wycofanie zmian należy kliknąć pole „Roll Back” po dwa kroć



Ostatnim krokiem będzie zweryfikowanie zmian

```
$ curl http://dotnet-dotnet-prod-student11.apps.lab19.example.com/ 2> /dev/null | grep "Linux\|ENVIRONMENT"
```

```
<h1> PRODUCTION ENVIRONMENT </h1>
```

```
<p>&copy; 2018 - LinuxPolska</p>
```

