



**POLITECHNIKA
RZESZOWSKA**
im. IGNACEGO ŁUKASIEWICZA

**SPRAWOZDANIE PROJEKTOWE
Z PRZEDMIOTU USŁUGI SIECIOWE W
BIZNESIE**

**Azure DevOps – Aplikacja mikroserwisowa z
wykorzystaniem Dockera**

**Zuzanna Kozek | 169800
FSO-DI**

Rzeszów, 2024

Spis treści

1. Cel projektu.....	3
2. Dlaczego DevOps?.....	3
2.1. Kluczowe praktyki w DevOps:.....	3
3. Pierwsze kroki z AzureDevOps	4
4. Aplikacja mikroserwisowa.....	10
4.1. Azure Aplikacja Kontenerowa.....	10
4.2. Realizacja aplikacji kontenerowej	12
4.3. Skalowalność w aplikacji kontenerowej.....	20
5. Podsumowanie	23

1. Cel projektu

Poniższy projekt ma na celu zapoznanie się z koncepcją DevOps, jej znaczeniem w pracy inżyniera oprogramowania oraz praktyczne zastosowanie narzędzi Azure DevOps w procesie tworzenia aplikacji. W szczególności projekt skupiał się na opracowaniu aplikacji mikroservisowej z wykorzystaniem technologii kontenerowych Docker, wdrożonej na platformie Azure.

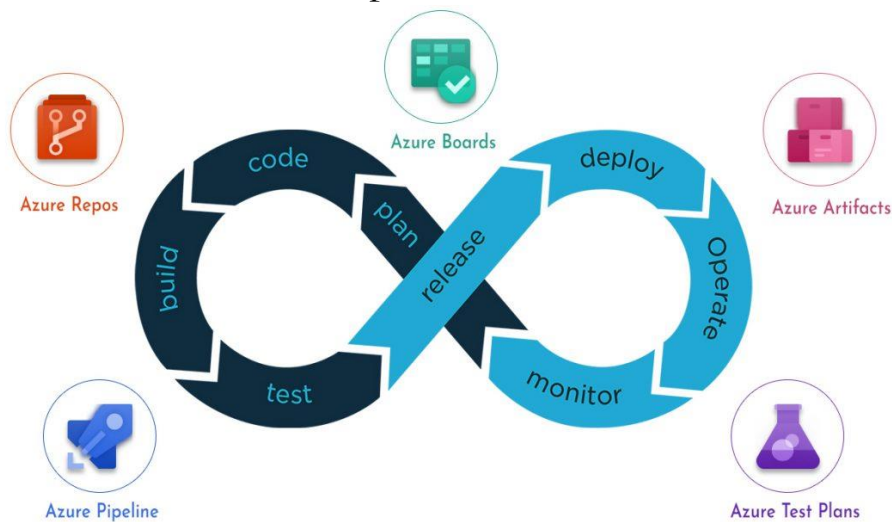
2. Dlaczego DevOps?

DevOps to skrót od „Development” (rozwój) i „Operations” (operacje). Jest to metodyka oraz kultura pracy, która łączy zespoły programistyczne (deweloperów) i operacyjne (administratorów systemów) w celu szybszego i bardziej niezawodnego dostarczania oprogramowania. DevOps jest odpowiedzią na tradycyjne problemy związane z izolacją tych dwóch funkcji, które często prowadziły do opóźnień, błędów i nieefektywności. DevOps jest zdolny do ciągłej integracji i ciągłego rozwoju, co oznacza, że tworzenie i wdrażanie aplikacji może być w pełni zautomatyzowane przy użyciu narzędzi do automatyzacji.

2.1. Kluczowe praktyki w DevOps:

- Continuous Integration (CI):
CI polega na częstym integrowaniu kodu zmienianego przez różnych deweloperów do głównego repozytorium. Każda integracja jest automatycznie testowana, co pozwala szybko wykrywać i naprawiać błędy.
- Continuous Delivery (CD):
CD rozszerza CI poprzez automatyzację procesu wdrażania kodu do środowisk produkcyjnych. Każda zmiana, która przejdzie pomyślnie przez fazę testów, może być automatycznie wdrożona na produkcję, co minimalizuje ryzyko błędów wdrożeniowych.
- Infrastructure as Code (IaC):
IaC to praktyka zarządzania infrastrukturą IT (serwery, sieci, bazy danych) za pomocą plików konfiguracyjnych, które można automatycznie wdrażać. Dzięki temu infrastruktura jest bardziej przewidywalna i łatwiejsza do zarządzania.
- Monitoring i Logowanie:
DevOps kładzie duży nacisk na monitorowanie aplikacji i infrastruktury w celu szybkiego wykrywania problemów i optymalizacji wydajności. Narzędzia do logowania i monitoringu pomagają w analizie danych i podejmowaniu świadomych decyzji.
- Automatyzacja:
Automatyzacja jest kluczowym elementem DevOps. Wszystkie powtarzalne i czasochłonne zadania, takie jak testowanie, wdrażanie, zarządzanie infrastrukturą, są automatyzowane, co pozwala zespołom skupić się na bardziej wartościowych działaniach.

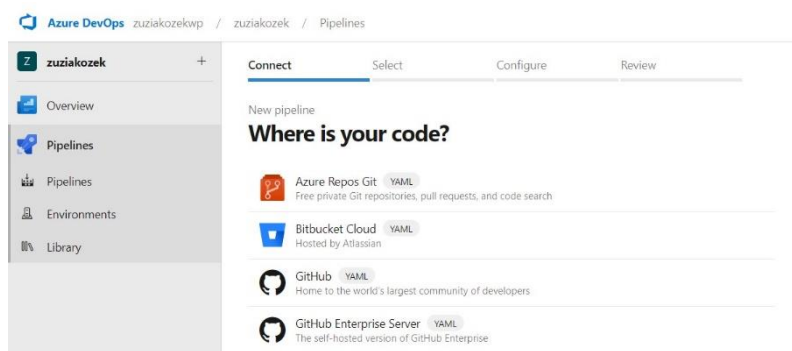
3. Pierwsze kroki z AzureDevOps



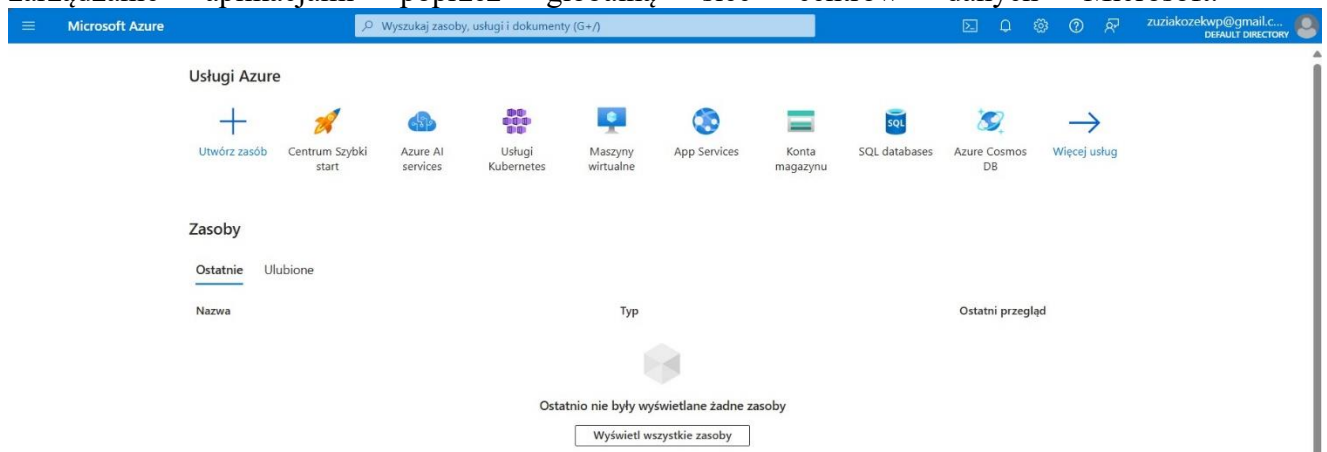
Azure DevOps to zestaw narzędzi i usług dostarczanych przez Microsoft, które umożliwiają zespołom programistycznym efektywne zarządzanie projektami, rozwijanie oprogramowania, testowanie oraz wdrażanie. Azure DevOps łączy w sobie najlepsze praktyki DevOps, co pozwala na szybsze i bardziej efektywne dostarczanie wysokiej jakości oprogramowania.

Aby zacząć pracę w usłudze AzureDevOps, należy zarejestrować się na stronie dev.azure.com. Dostęp jest darmowy przez rok:

The screenshot shows the registration process for a new Azure DevOps account. On the left, the 'Your profile' section contains a form with the following fields: 'Country/Region' (set to Poland), 'First name' (Zuzanna), 'Middle name (Optional)' (empty), 'Last name' (Kozek), 'Email address' (zuziakozekwp@gmail.com), and 'Phone' (with an example XXX XXX XXX). Below the form are checkboxes for 'Use a different phone number to verify your identity.' and buttons for 'Text me' and 'Call me'. On the right, the 'Create your Azure free account' section highlights benefits: 'Popular services free for 12 months', '55+ services always free', and 'USD200 credit to use in your first 30 days'. It also includes a 'No automatic charges' warning.



Następnym krokiem do utworzenia aplikacji jest połączenie się z usługą chmurową Microsoft Azure. Dostarcza ona szeroki zakres usług umożliwiających tworzenie, wdrażanie i zarządzanie aplikacjami poprzez globalną sieć centrów danych Microsoft.



W planie bezpłatnym otrzymujemy 60 minut pracy procesora:

Popularne plany	
Bezpłatna F1	Uwzględniono 60 minut procesora CPU na dzień
Udostępniona D1	Uwzględniono 60 minut procesora CPU na dzień
Podstawowa B1 54,75 USD/miesiąc (szacowane)	ACU: 100, Pamięć: 1.75 GB, Procesor wirtualny: 1
Standard S1 73,00 USD/miesiąc (szacowane)	ACU: 100, Pamięć: 1.75 GB, Procesor wirtualny: 1
Premium (wersja 3) P0V3 172,86 USD/miesiąc (szacowane)	ACU: 195, Pamięć: 4 GB, Procesor wirtualny: 1
Premium (wersja 3) P1V3 271,41 USD/miesiąc (szacowane)	ACU: 195, Pamięć: 8 GB, Procesor wirtualny: 2
Premium (wersja 3) P1MV3 298,56 USD/miesiąc (szacowane)	ACU: 195, Pamięć: 16 GB, Procesor wirtualny: 2
Izolowana V2 I1V2 408,80 USD/miesiąc (szacowane)	ACU: 195, Pamięć: 8 GB, Procesor wirtualny: 2
Bezpłatna F1 (Współużytkowana infrastruktura)	

Tworzenie aplikacji:

[Strona główna](#) > [Create a resource](#) >

Utwórz aplikację internetową

rygorystyczne wymagania dotyczące wydajności, skalowalności, zabezpieczeń i zgodności, używając w pełni zarządzanej platformy do konserwacji infrastruktury. [Dowiedz się więcej](#)

Szczegóły projektu

Wybierz subskrypcję, aby zarządzać wdrożonymi zasobami i kosztami. Użyj grup zasobów jak folderów, aby organizować wszystkie Twoje zasoby i zarządzać nimi.

Subskrypcja * [?](#) Azure subscription 1

Grupa zasobów * [?](#) (Nowy) New [Utwórz nowy](#)

Szczegóły wystąpienia

Nazwa * projektazure [✓](#)
.azurewebsites.net

Publikuj * ☒ Kod ☐ Container ☐ Statyczna aplikacja internetowa

Stos środowiska uruchomieniowego * .NET 6 (LTS)

System operacyjny * ☐ Linux ☒ Windows

Region * Poland Central

[i](#) Nie możesz znaleźć planu usługi App Service? Spróbuj użyć innego regionu lub wybierz środowisko App Service Environment.

Ukończenie tworzenia zasobu na aplikację:

✓ Wdrożenie zostało ukończone



Nazwa wdrożenia: Microsoft.Web-WebApp-Portal-0b1e3a07-92e5
Subskrypcja: Azure subscription 1
Grupa zasobów: New

Godzina rozpoczęcia: 24.05.2024, 21:55:14

Identyfikator korelacji: 92452d64-1275-4b57-8f57-27943c6821a2 [📄](#)

▼ Szczegóły wdrożenia

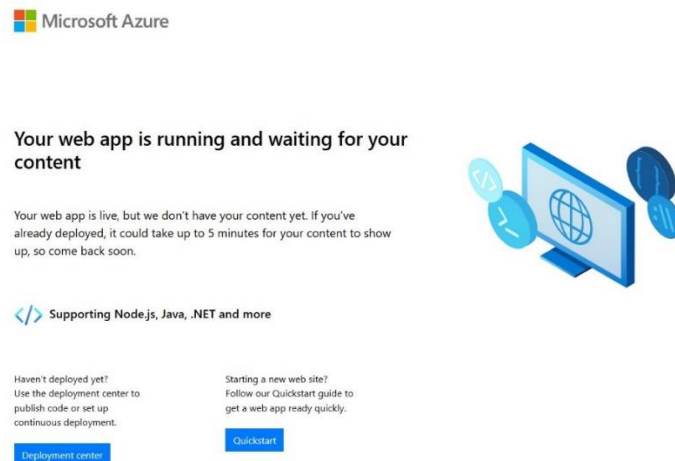
^ Kolejne kroki

[Zarządzaj wdrożeniami aplikacji](#) Proponowane

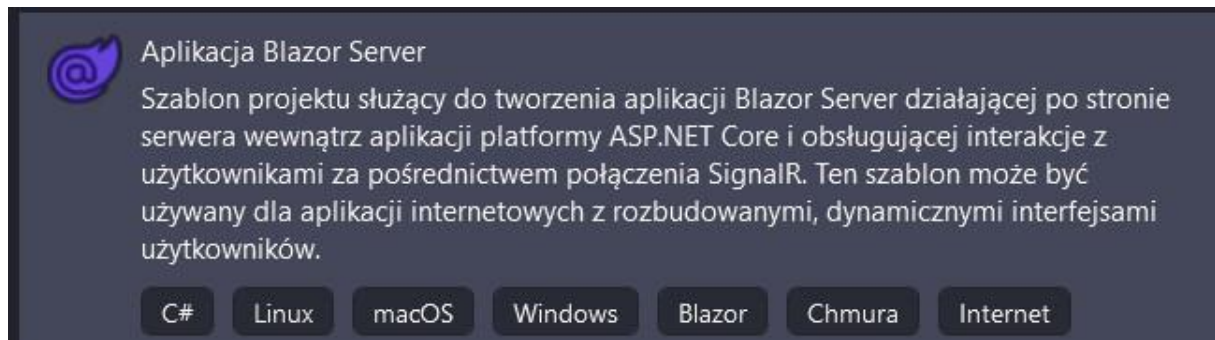
[Chroń aplikację przy użyciu uwierzytelniania](#) Proponowane

[Przejdź do zasobu](#)

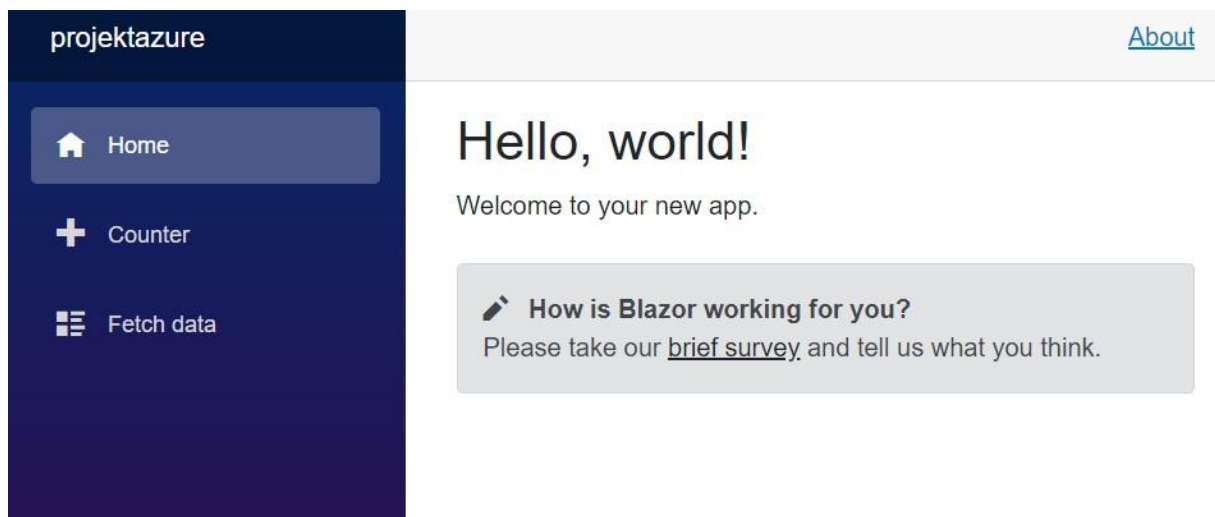
Domyślny wygląd aplikacji po uruchomieniu generowany przez Microsoft:



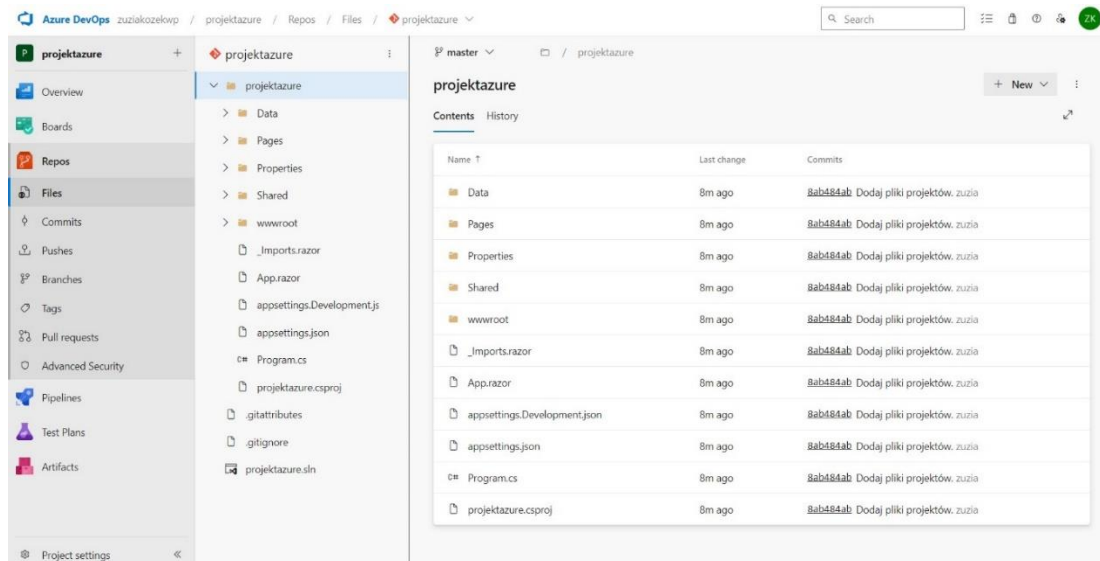
Do stworzenia struktury aplikacji skorzystałam z Aplikacji Blazor Server



Wystartowana aplikacja z VisualStudio:



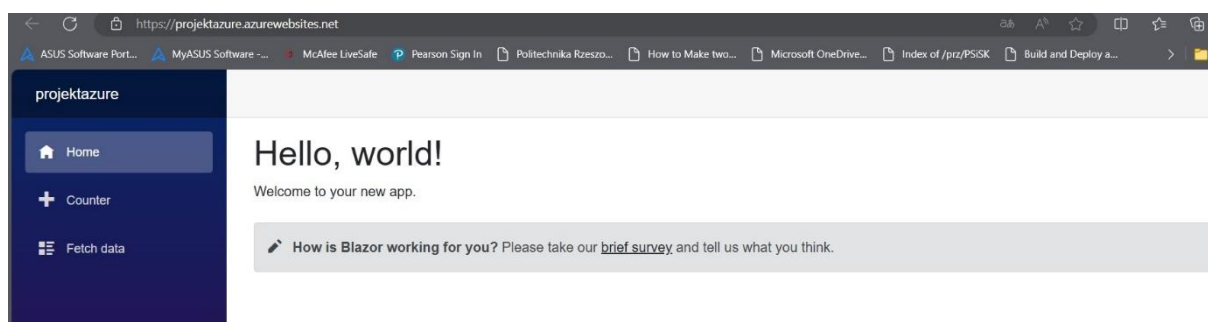
Dzięki połączeniu z usługą Git, możliwe jest „wypchanie” kodu do usługi Azure DevOps Repos:



W usłudze chmurowej, w zakładce centrum wdrażania należy połączyć się z repozytorium:



Od teraz nasza aplikacja działa na domenie azurewebsites:

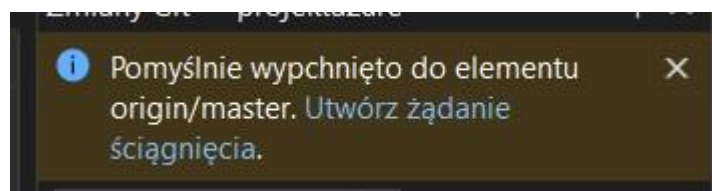


Czy zmiany w kodzie są widoczne?

W kodzie źródłowym wprowadziłam zmiany:

```
Index.razor*  [X]
1  @page "/"
2
3  <PageTitle>Index</PageTitle>
4
5  <h1>Zmiana w kodzie!</h1>
6
7  Welcome to your new app.
8
9  <SurveyPrompt Title="How is Blazor working for you?" />
10
```

Następnie, poprzez usługę Git, zaktualizowałam kody:



W zakładce Dzienniki znajdziemy szczegóły z działania aplikacji. Po aktualizacji z nadaną nazwą „kod”, taki sam komunikat znajduje się w dzienniku:

Ustawienia

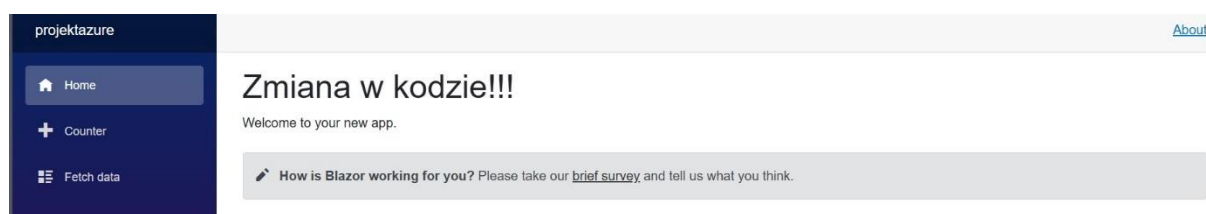
Dzienniki

Poświadczenia protokołu FTPS

Odśwież

Usuń

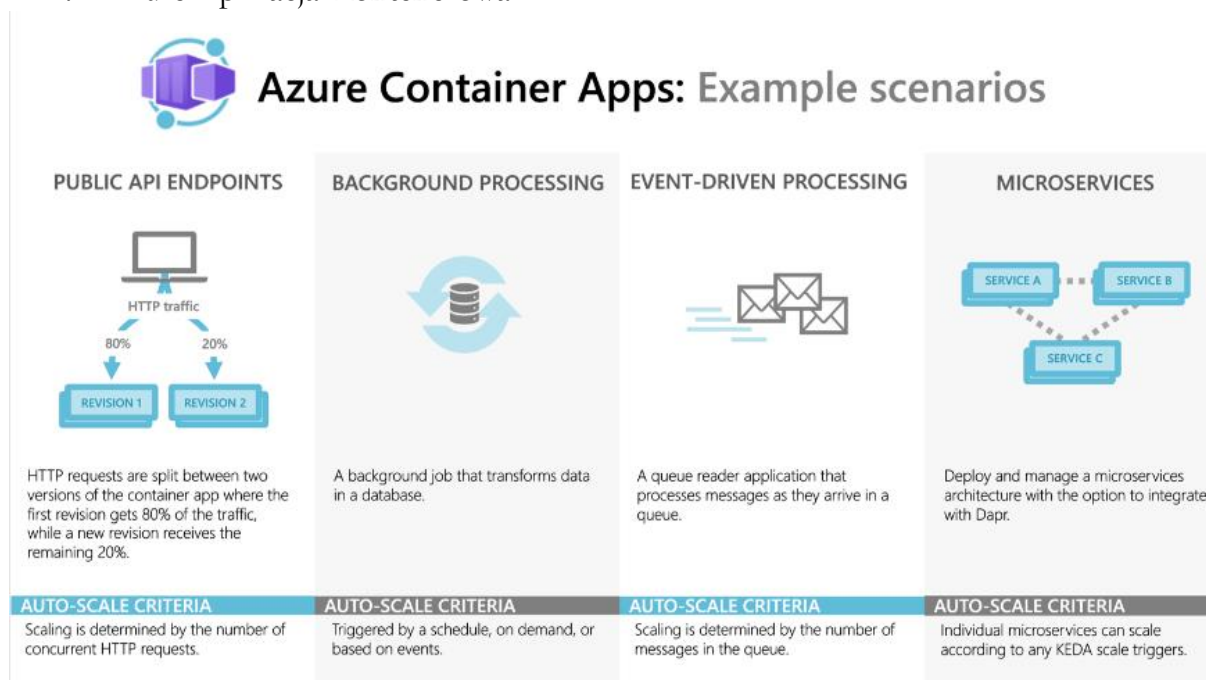
▼	Czas	Identyfikator ...	Autor zatwierdzenia	Stan	Komunikat
▼	Friday, May 24, 2024 (2)				
	11:00:25 PM +02:00	9d8bbc6	Zuzanna Kozek	Powodzenie (Aktywne)	kod
	10:54:46 PM +02:00	8ab484a	zuzia	Powodzenie	Dodaj pliki projektów.



4. Aplikacja mikroserwisowa

Technika mikroserwisowa polega na tworzeniu aplikacji jako zbioru małych, niezależnych usług, które komunikują się ze sobą przez dobrze zdefiniowane interfejsy. Każda usługa (mikroserwis) jest wdrażana, skalowana i zarządzana niezależnie, co pozwala na większą elastyczność, skalowalność i odporność aplikacji. W moim projekcie stworzyłam aplikację kontenerową, wykorzystując Docker do pakowania mikroserwisów w lekkie, przenośne kontenery, które następnie wdrożyłam na platformie Azure.

4.1. Azure Aplikacja Kontenerowa



Azure Container Apps to bezserwerowa platforma, która umożliwia utrzymanie mniejszej infrastruktury i oszczędność kosztów podczas uruchamiania konteneryzowanych aplikacji. Zamiast martwić się o konfigurację serwera, aranżację kontenera i szczegóły wdrożenia, usługa Container Apps udostępnia wszystkie aktualne zasoby serwera wymagane do zapewnienia stabilności i bezpieczeństwa aplikacji.

Typowe zastosowania usługi Azure Container Apps obejmują:

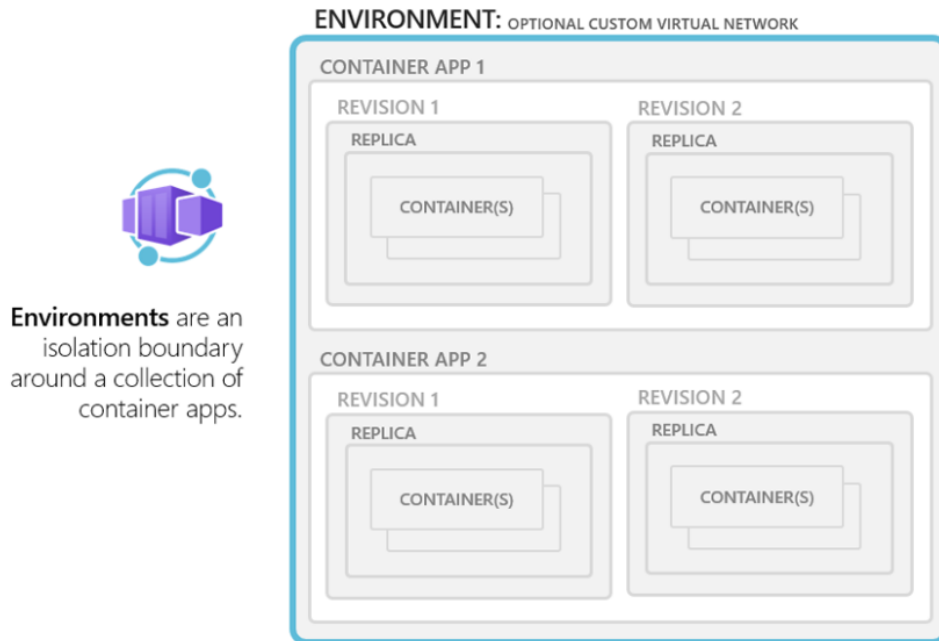
- Wdrażanie punktów końcowych interfejsu API
- Hostowanie zadań przetwarzania w tle
- Obsługa przetwarzania sterowanego zdarzeniami
- Uruchamianie mikrousług

Ja zajmę się w moim projekcie mikrousługami.

Ponadto aplikacje oparte na usłudze Azure Container Apps mogą dynamicznie skalować na podstawie następujących cech:

- Ruch HTTP
- Przetwarzanie sterowane zdarzeniami
- Obciążenie procesora CPU lub pamięci

Tą tematyką zajmę się w dalszej części mojego projektu.



Środowisko Container Apps to bezpieczna granica wokół co najmniej jednej aplikacji i zadań kontenera. Środowisko uruchomieniowe usługi Container Apps zarządza poszczególnymi środowiskami, obsługując uaktualnienia systemu operacyjnego, operacje skalowania i równoważenie zasobów.

4.2. Realizacja aplikacji kontenerowej

Pierwszą aplikacją kontenerową jaką stworzyłam była aplikacja frontend. Będzie ona zawierała odwołania do drugiej aplikacji, co pokażę w dalszym etapie.

[...](#) > [AplikacjaKontenerowa](#) > [Marketplace](#) > [Aplikacja kontenera](#) >

Utwórz element Aplikacja kontenera

Zakończono pomyślnie

[Podstawowe informacje](#) [Kontener](#) [Powiązania](#) [Tagi](#) [Przejrzyj i utwórz](#)

Szczegóły projektu

Subskrypcja	Azure subscription 1
Grupa zasobów	AplikacjaKontenerowa
Nazwa	frontend

Środowisko usługi Container Apps (nowość)

Region	polandcentral
Środowisko usługi Container Apps	app-env
Obszar roboczy usługi Log Analytics (nowość)	app-logs
Sieć wirtualna	Domyślnie
Nadmiarowość strefowa	Wyłączone

Kontener

Źródło obrazu	Szybki start
Obraz	mcr.microsoft.com/k8se/quickstart:latest
Typ profilu obciążenia	Zużycie
Liczba rdzeni procesora CPU	0.25
Rozmiar pamięci (Gi)	0.5

✓

Wdrożenie zostało ukończzone

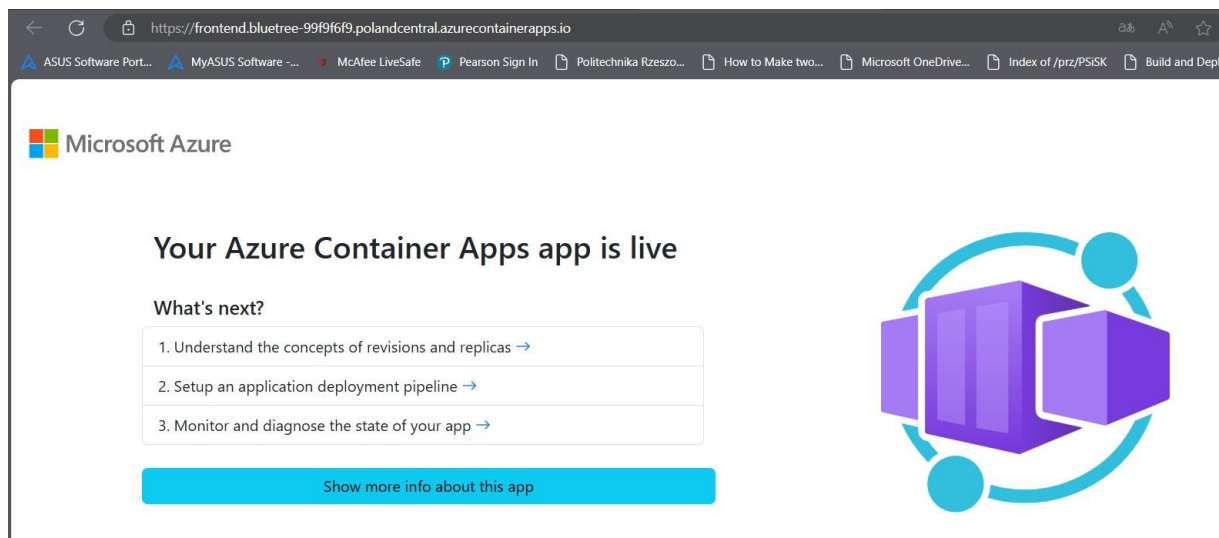
 Nazwa wdrożenia : Microsoft.App-ContainerApp-Portal-87a...
Subskrypcja : Azure subscription 1
Grupa zasobów : AplikacjaKontenerowa
Czas rozpoczęcia : 25.05.2024, 11:06:00
Identyfikator korelacji : 09289934-172e-4e2b-ae59-990...

> Szczegóły wdrożenia

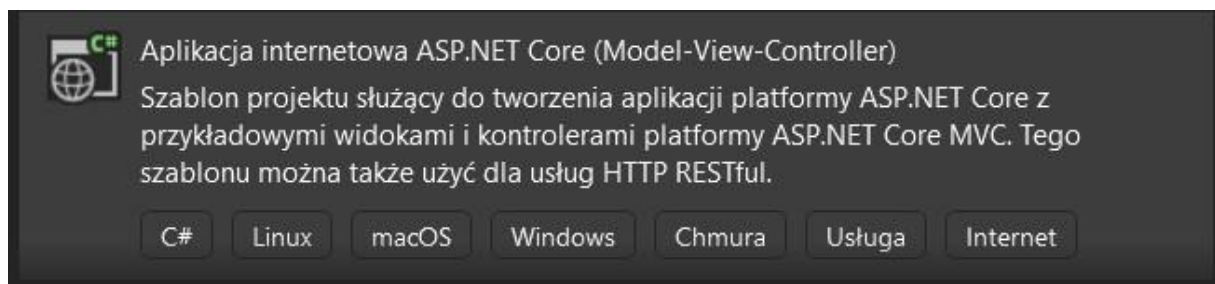
✓ Kolejne kroki

Przejdź do zasobu

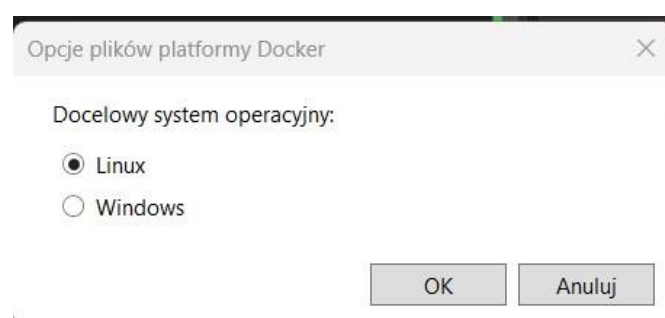
Zanim wprowadziłam zmiany w aplikacji, to pod adresem URL znajduje się domyślna strona:



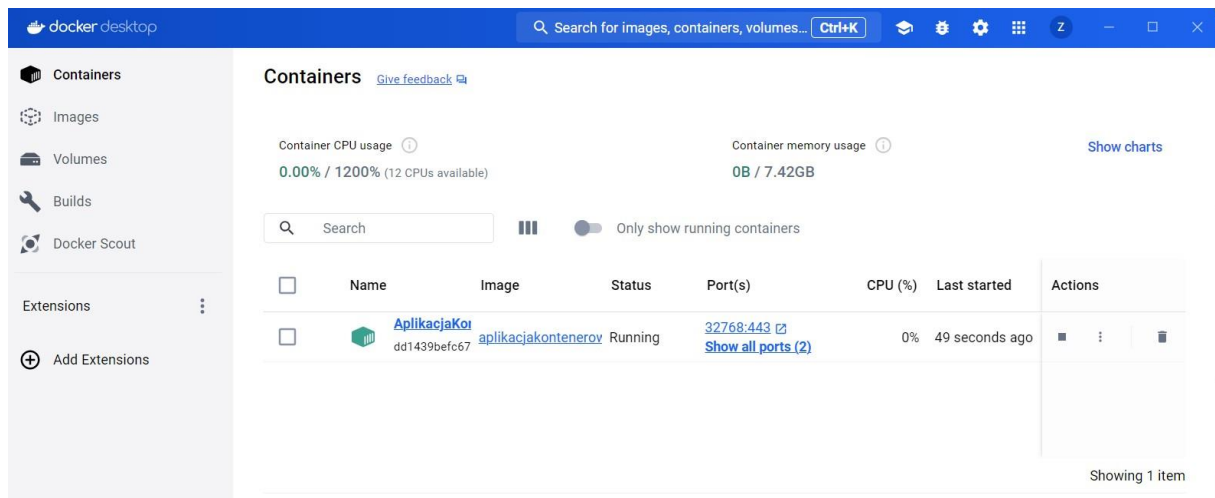
Przeszłam do stworzenia szablonu mojej strony. Szablon zawiera prosty wzór zadań do zrobienia (ang. "to do"). Wykorzystałam do tego projekt ASP.NET Core aplikacja:



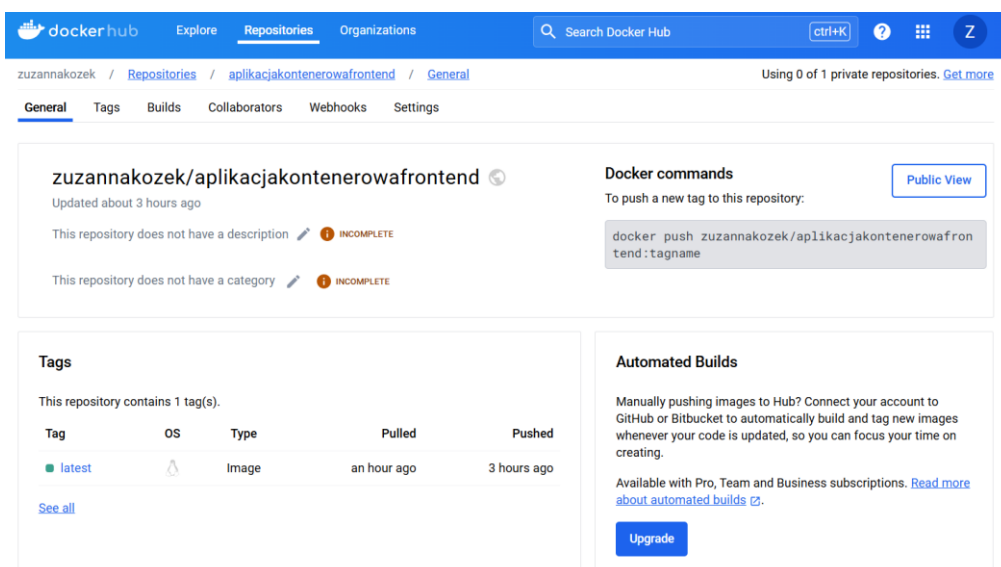
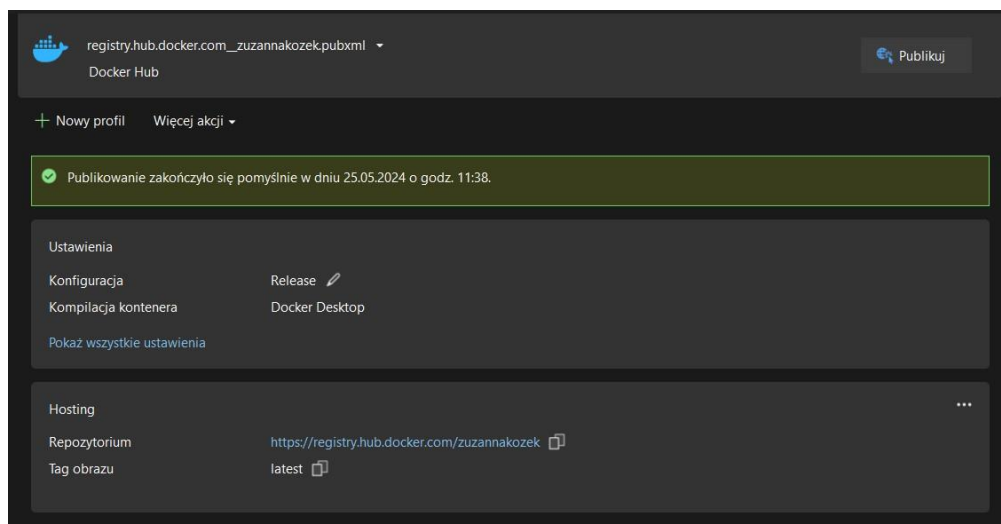
Po załadowaniu plików zajęłam się połączeniem obsługi platformy Docker. Jako docelowy system operacyjny wybrałam Linux:



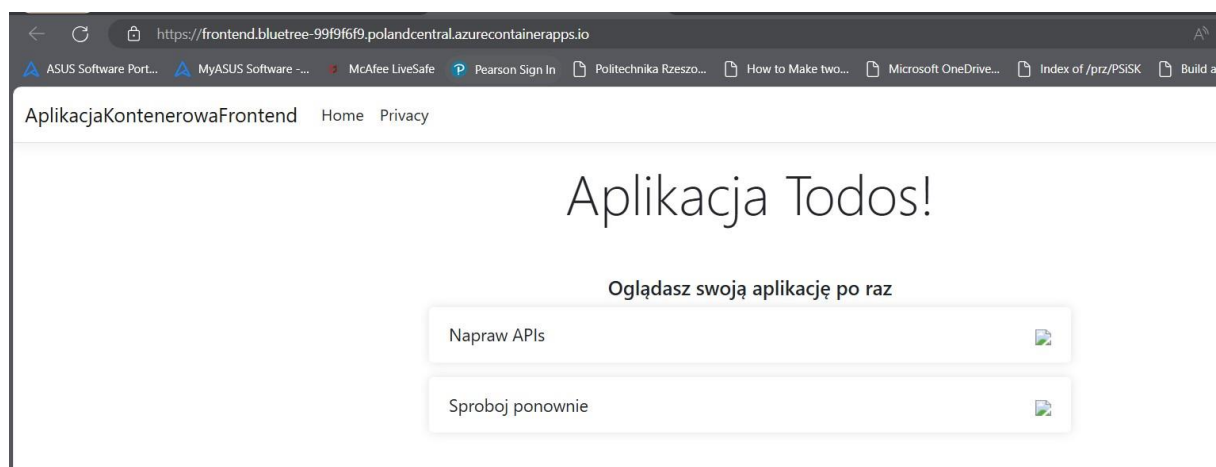
Po dodaniu i uruchomieniu programu Docker Desktop, aplikacja została dodana jako kontener:



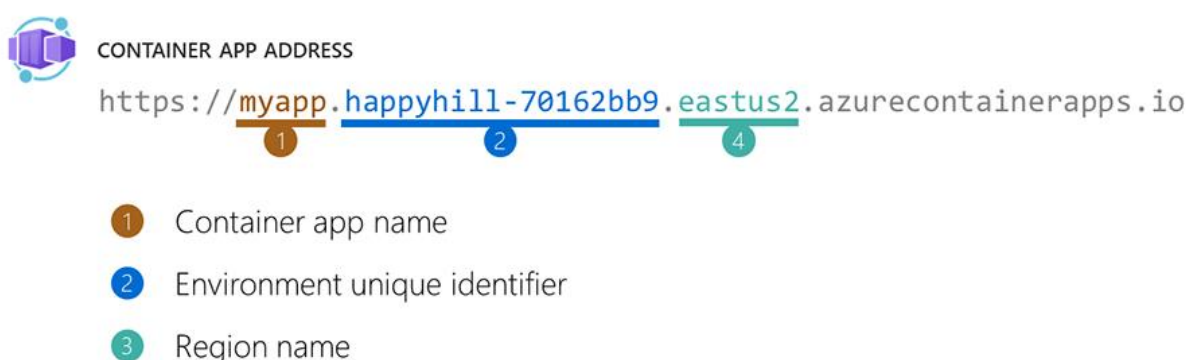
Następnie opublikowałam mój kontener do Docker Hub:



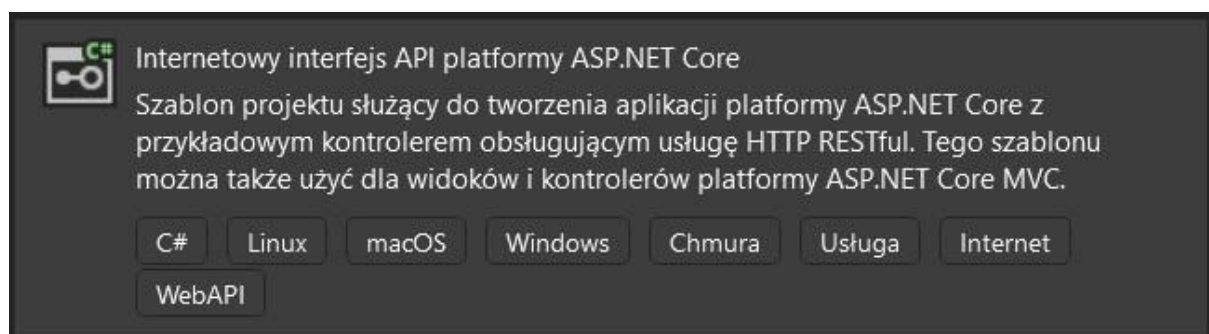
W kolejnym kroku, na platformie Microsoft Azure utworzyłam nową aplikację kontenerową, która zawiera źródło obrazu z Docker Hub. W wyniku otrzymałam aplikację kontenerową zawierającą obraz kontenera Docker z kodem źródłowym w Visual Studio:



Następny kontener, który będzie zawierał backend mojej aplikacji, stworzyłam jako środowisko wewnętrzne. Środowiska wewnętrzne nie mają publicznych punktów końcowych i są wdrażane z wirtualnym adresem IP (VIP) mapowanym na wewnętrzny adres IP. Wewnętrzny punkt końcowy to wewnętrzny moduł równoważenia obciążenia platformy Azure (ILB), a adresy IP są wydawane z listy prywatnych adresów IP niestandardowej sieci wirtualnej.



Do utworzenia drugiej aplikacji skorzystałam z Internetowego interfejsu API:



W kontrolerze aplikacji backend umieściłam zawartość mojej strony:

```
using Microsoft.AspNetCore.Mvc;

namespace backend.Controllers
{
    Odwołania: 5
    public class FoodItem
    {
        Odwołania: 3
        public int Id { get; set; }
        Odwołania: 3
        public string Name { get; set; }
        Odwołania: 3
        public string Description { get; set; }
        Odwołania: 3
        public string Price { get; set; }
    }

    [ApiController]
    [Route("[controller]")]
    Odwołania: 0
    public class BackendController : ControllerBase
    {
        [HttpGet]
        Odwołania: 0
        public IActionResult Get()
        {
            List<FoodItem> foodItems = new List<FoodItem>
            {
                new FoodItem { Id = 1, Name = "Pizza", Description = "Pepperoni pizza", Price = "40zł"},
                new FoodItem { Id = 2, Name = "Burger", Description = "Cheeseburger", Price = "30zł" },
                new FoodItem { Id = 3, Name = "Sushi", Description = "California roll", Price = "25zł" }
            };

            return Ok(foodItems);
        }
    }
}
```


Natomiast we wcześniej utworzonym frontend umieściłam jedynie odwołania do backendu:

```
using System.Diagnostics;

namespace AplikacjaKontenerowaFrontend.Controllers
{
    1 odwołanie
    public class FoodItem
    {
        Odwołania: 0
        public int Id { get; set; }
        Odwołania: 0
        public string Name { get; set; }
        Odwołania: 0
        public string Description { get; set; }
        Odwołania: 0
        public string Price { get; set; }
    }

    Odwołania: 3
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        Odwołania: 0
        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }


        Odwołania: 0
        public async Task<IActionResult> Index()
        {
            HttpClient client = new HttpClient();
            var result = await client.GetAsync("https://backend.internal.bluetree-99f9f6f9.polandcentral.azurecontainerapps.io/backend");
            var text = await result.Content.ReadAsStringAsync();
            ViewBag.backend = JsonConvert.DeserializeObject<List<FoodItem>>(text);

            return View();
        }

        Odwołania: 0
        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        Odwołania: 0
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
```

Analogicznie opublikowałam moją nową aplikację backend do Docker Hub:

 dockerhub

ExploreRepositoriesOrganizations

ctrl+K

?

Z


zuzannakozek / Repositories / backend / General


Using 0 of 1 private repositories. [Get more](#)

GeneralTagsBuildsCollaboratorsWebhooksSettings

zuzannakozek/backend

Updated about 3 hours ago

This repository does not have a description  INCOMPLETE

This repository does not have a category  INCOMPLETE

Docker commands

To push a new tag to this repository:

```
docker push zuzannakozek/backend:tagname
```

Public View

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	linux	Image	an hour ago	3 hours ago

[See all](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

Upgrade

17

W sekcji adresu umieściłam adres do aplikacji backend z dodanym słowem internal. Zatem jest to aplikacja niewidoczna dla użytkowników internetowych.

```
var result = await client.GetAsync("https://backend.internal.bluetree-99f9f6f9.polandcentral.azurecontainerapps.io/backend");
```

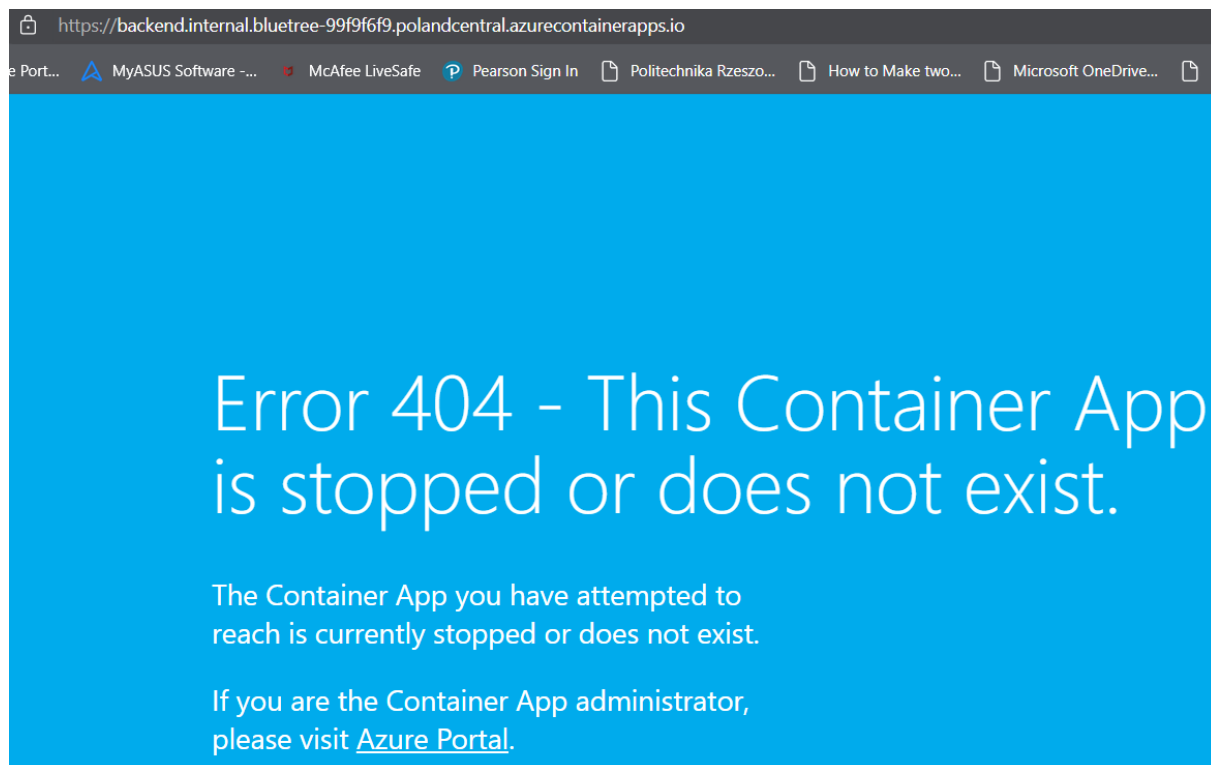
Następnie stworzyłam aplikację kontenerową backend w Microsoft Azure:

The screenshot shows the 'Utwórz element Aplikacja kontenera' (Create container app element) form in the Azure portal. The breadcrumb navigation at the top reads: '... > AplikacjaKontenerowa > Marketplace > Aplikacja kontenera >'. The form is titled 'Utwórz element Aplikacja kontenera' with a close button (X) in the top right corner. Below the title, there is a link 'zasoby lub utwórz je teraz. Dowiedz się więcej' (resources or create them now. Learn more). The 'Szczegóły projektu' (Project details) section contains three fields: 'Subskrypcja *' (Subscription) with a dropdown menu showing 'Azure subscription 1'; 'Grupa zasobów *' (Resource group) with a dropdown menu showing 'AplikacjaKontenerowa' and a link 'Utwórz nowy' (Create new); and 'Nazwa aplikacji kontenera *' (Container app name) with a text input field containing 'backend'. The 'Środowisko usługi Container Apps' (Container Apps service environment) section includes a checkbox 'Pokaż środowiska we wszystkich regionach' (Show environments in all regions) which is unchecked, a 'Region *' dropdown menu showing 'Poland Central', and a 'Środowisko usługi Container Apps *' dropdown menu showing 'app-env (AplikacjaKontenerowa)' with a link 'Utwórz nowy' (Create new). At the bottom of the form, there are three buttons: 'Przejrzyj i utwórz' (View and create) in blue, 'Wstecz <' (Back) in grey, and 'Dalej : Kontener >' (Next : Container) in grey.

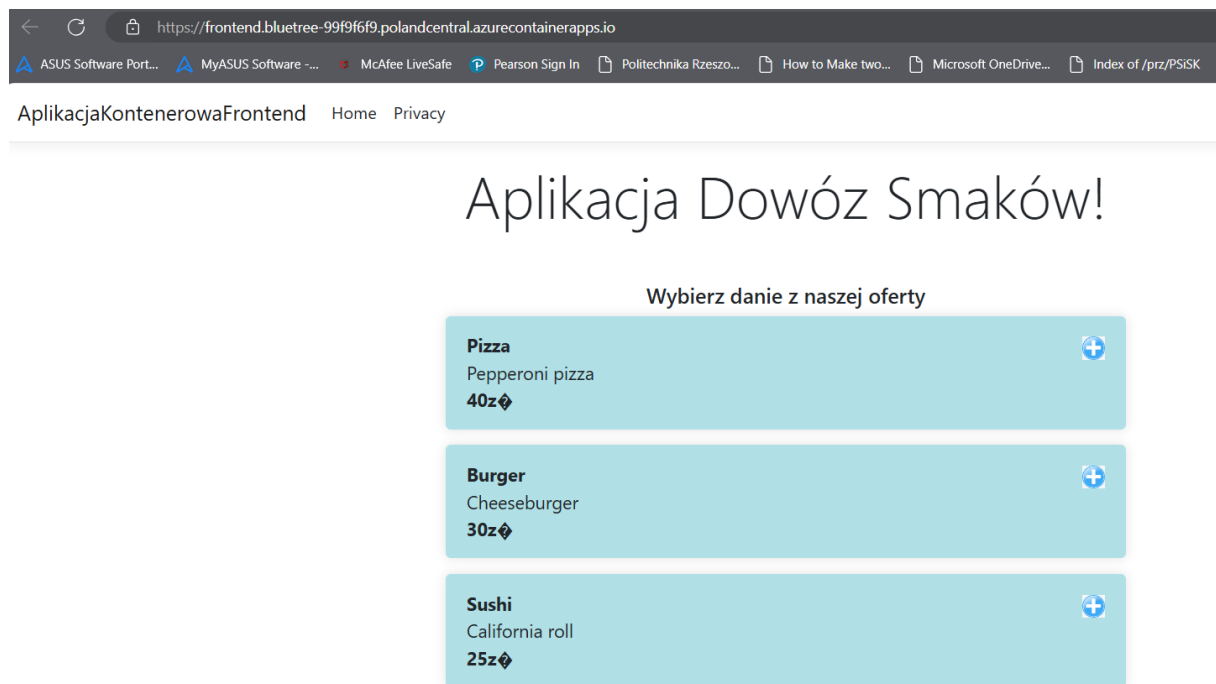
Aby utworzyć komunikację pomiędzy kontenerami, wybrałam wspólne środowisko usługi.

Aby zablokować ruch internetowy, wybrałam opcję : Ograniczone do środowiska usługi Azure Container Apps.

Po utworzeniu aplikacji backend, spróbowałam uruchomić jej URL. Wynikiem był error co jest pożądanym zjawiskiem, ponieważ ten kontener jest wewnętrzny:



Aby wyświetlić moją usługę uruchomiłam adres URL aplikacji frontend:



Strona działa i zawiera listę dań dostępnych w ofercie lokalu, która została utworzona w aplikacji backend. Jest to możliwe dzięki połączeniu obu kontenerów.

4.3. Skalowalność w aplikacji kontenerowej

Aplikacje kontenerowe obsługują skalowanie poziome. Gdy aplikacja kontenerowa skaluje się, nowe instancje aplikacji kontenerowej są tworzone na żądanie. Instancje te nazywane są replikami. Podczas pierwszego tworzenia aplikacji kontenera reguła skalowania jest ustawiona na zero.

Rodzaje skalowania:

Ruch HTTP: Skalowanie na podstawie liczby jednoczesnych żądań HTTP do wersji.

Sterownik zdarzeń: Wyzwalacze oparte na zdarzeniach, takie jak komunikaty w usłudze Azure Service Bus.

CPU/Pamięć: Skalowanie w oparciu o ilość procesora CPU lub pamięci zużywanej przez replikę.

Do mojej aplikacji skorzystałam ze skalowania http:

Dodawanie reguły skalowania

Szczegóły reguły skalowania

Nazwa reguły *

http-regula

Typ * ⓘ

Skalowanie HTTP

Współbieżne żądania *

20

Maksymalną ilość replik ustawiłam na 5:

Skalowanie

Ustawienie reguły skalowania

Minimalna/maksymalna liczba replik0 - 5

Reguła skalowania

Nazwa ↑	Typ ↑
http-regula	Skalowanie HTTP

Zanim podjęłam się wysyłania żądań http, to liczba replik mojej aplikacji wynosiła 1:

frontend | Konsola

Aplikacja kontenera

Wyszukaj

Odśwież

Ruch przychodzący

Deployment

Domeny niestandardowe

Dapr

Wybierz wystąpienie kontenera, aby połączyć się z konsolą.

Replika

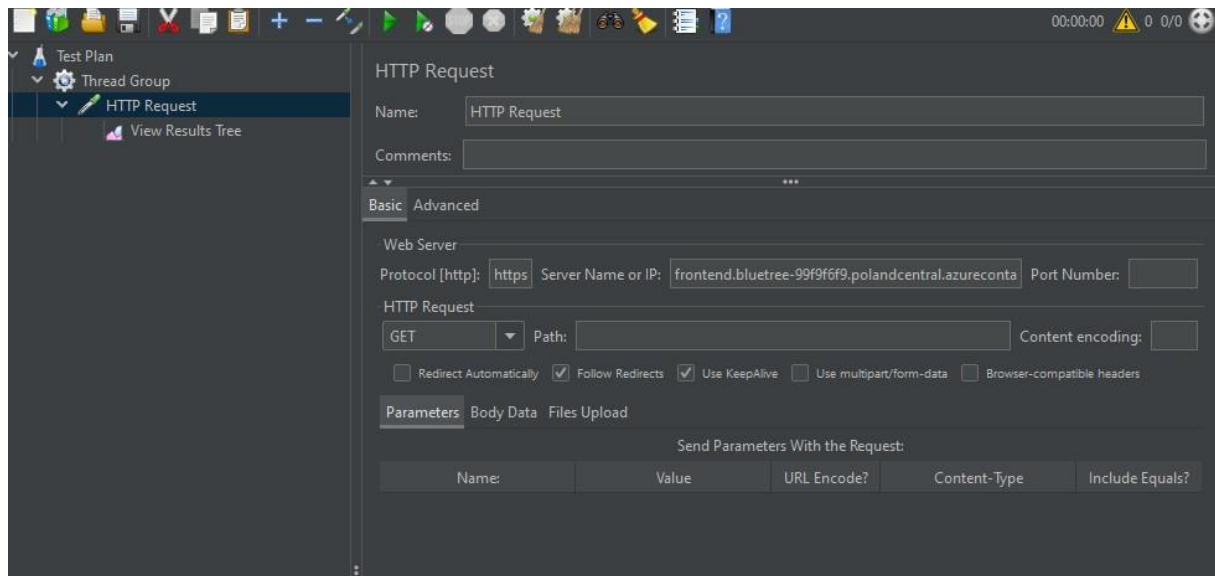
frontend--1paynay-6bb9cd84bc-lbkcp

Kontener

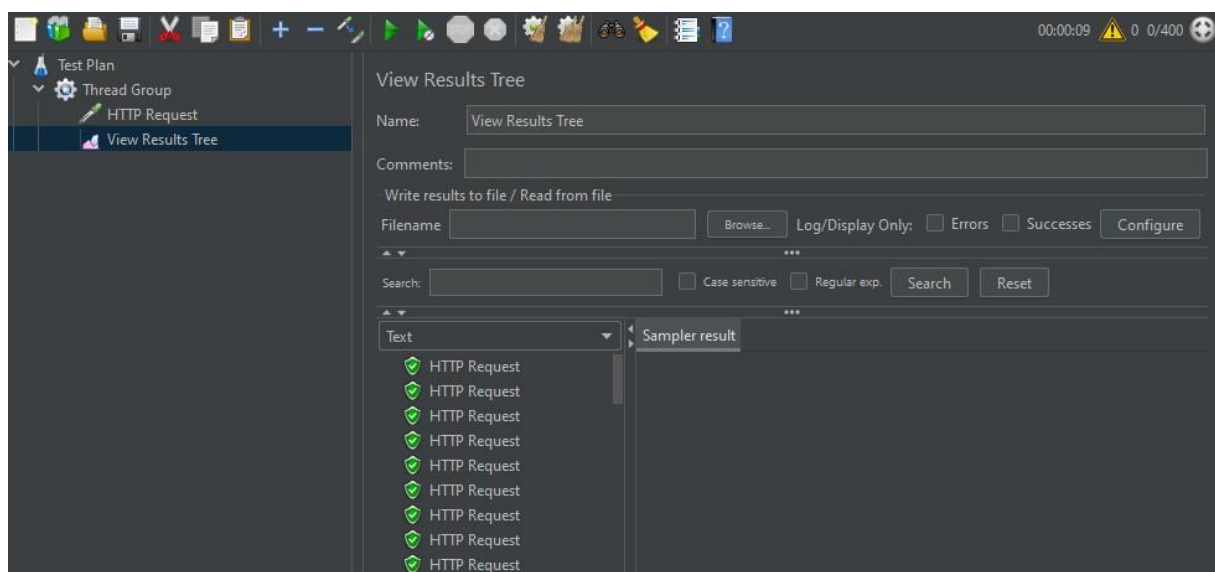
Połącz ponownie

21

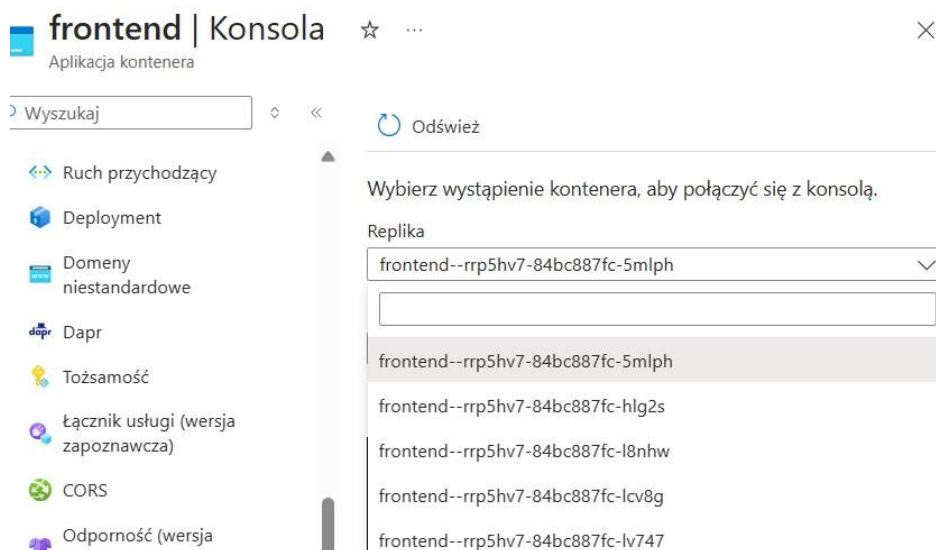
Do wysyłania żądań do mojej aplikacji frontend skorzystałam z programu Apache JMeter:



Wyniki w Listenerze:



Po zakończeniu przesyłania żądań w sekcji konsoli, znajdziemy wiadomość, że ilość replik mojej aplikacji wynosi 5, czyli maksymalna ilość:



5. Podsumowanie

Projekt zaprezentował praktyczne zastosowanie narzędzi Azure DevOps do tworzenia i zarządzania aplikacjami mikroserwisowymi z wykorzystaniem Dockera. Implementacja kontenerów frontend i backend, ich publikacja oraz konfiguracja skalowalności pokazuje, jak efektywnie zarządzać nowoczesnymi aplikacjami w chmurze. Realizacja tego projektu dostarczyła cennych umiejętności i doświadczenia, które mogą być bezpośrednio zastosowane w pracy analityka danych. Automatyzacja procesów, skalowalność, integracja z różnymi źródłami danych, monitorowanie, logowanie oraz efektywne zarządzanie zasobami i zespołem to kluczowe aspekty, które zwiększają wartość i efektywność pracy analityka danych.