

Laboratorium Azure App Service

Kamil Makarowski, Filip Maciąg

Spis Treści:

1. Wymagania
2. Utworzenie Azure App Service Plan
3. Wdrożenie Azure Web App Service (HTML & FTP)
 - 3.1. Utworzenie Azure Web App Service
 - 3.2. Wdrożenie pliku HTML przy użyciu FTP
4. Wdrożenie Azure Web App Service (.NET & Visual Studio 2022)
5. Wdrożenie Azure Web App Service (React & Visual Studio Code)
6. Komunikacja między UI & API - CORS & AppSettings
7. Podstawy CI/CD
 - 7.1. Deployment Center
 - 7.2. Wdrożenie z testami
8. Deployment Slots
9. OBOWIĄZKOWE NA KONIEC ZAJĘĆ

1.Wymagania

- Dostęp do Azure Portal + aktywna subskrypcja ze środkami
- Zainstalowany pakiet [npm](#)
- Zainstalowane Visual Studio 2022
- Zainstalowane Visual Studio Code
- Filezilla (lub inny klient FTP)
- .NET SDK

2. Utworzenie Azure App Service Plan

- Zanim utworzymy Azure App Service Plan należy utworzyć **Resource Group**, w której będziemy trzymać nasze zasoby przygotowane w ramach laboratorium
 - Name: N/A (dobrą praktyką jest używanie nazw, które coś znaczą w obrębie zasobów)
 - Region: (Europe) Poland Central (region może być dowolny, ważne aby być konsekwentnym)
- Następnie należy przejść do utworzonej Resource Group i utworzyć nowy zasób – Azure App Service Plan:

Azure-App-Service-Lab-KM Resource group

Search « + Create Manage view Delete resource group Refresh Export to CSV Open query Assign tags

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Deployments

Security

Deployment stacks

Policies

Properties

Locks

Cost Management

Cost analysis

Cost alerts (preview)

Budgets

Advisor recommendations

Monitoring

Home > PD - RIP RDC - MSDN - 01 | Resource groups > Azure-App-Service-Lab-KM >

Essentials

Subscription (move) : PD - RIP RDC - MSDN - 01 Deployments : No deployments

Subscription ID : 6168bc6b-a39f-4430-9e73-a36f5d9cd09f Location : Poland Central

Tags (edit) : Add tags

Resources Recommendations

Filter for any field... Type equals all Location equals all Add filter

Showing 0 to 0 of 0 records. Show hidden types

Name ↑↓ Type ↑↓

No resources match your filters

Try changing or clearing your filters.

Create resources Clear filters

Marketplace

Get Started

Service Providers

Management

Private Marketplace

Private Offer Management

My Marketplace

Favorites

My solutions

Recently created

Private plans

Categories

IT & Management Tools

App Service Plan

Pricing : All Operating System

Azure benefit eligible only Azure services only

Showing 1 to 20 of 49 results for 'App Service Plan': Clear search

App Service Plan

Microsoft

Azure Service

An App Service plan represents a set of features and capacity that you can share across multiple apps in Azure App Service

Create

WordPress App Service with MySQL in-app

Frameworkx Technology

Azure Application

Simple WordPress app service with MySQL in-app

Price varies

Create

Strapi on Azure App Service

QuickDeploy

Azure Application

Strapi on Azure App Service with Azure Database for MySQL & Azure Storage



Price varies

Create

- Name: N/A
- Operating System: Windows (może być Linux, przy czym do zadania 8 wymagany jest Windows)
- Region: Taki jak Resource Group
- Pricing Tier: Standard S1 (Najtańszy tier, który umożliwia skonfigurowanie Deployment Slots)

UWAGA: Dla zwykłego developmentu zalecane jest zaczynać od tier **Free F1**, który jest darmowy, przy czym należy pamiętać, że taka instancja Azure Service Planu może być utworzona tylko jedna na cały region.

- Weryfikacja:
 - Zasób powinien być widoczny w Resource Group
 - Settings -> Apps powinno być puste

 Delete
  Send us your feedback

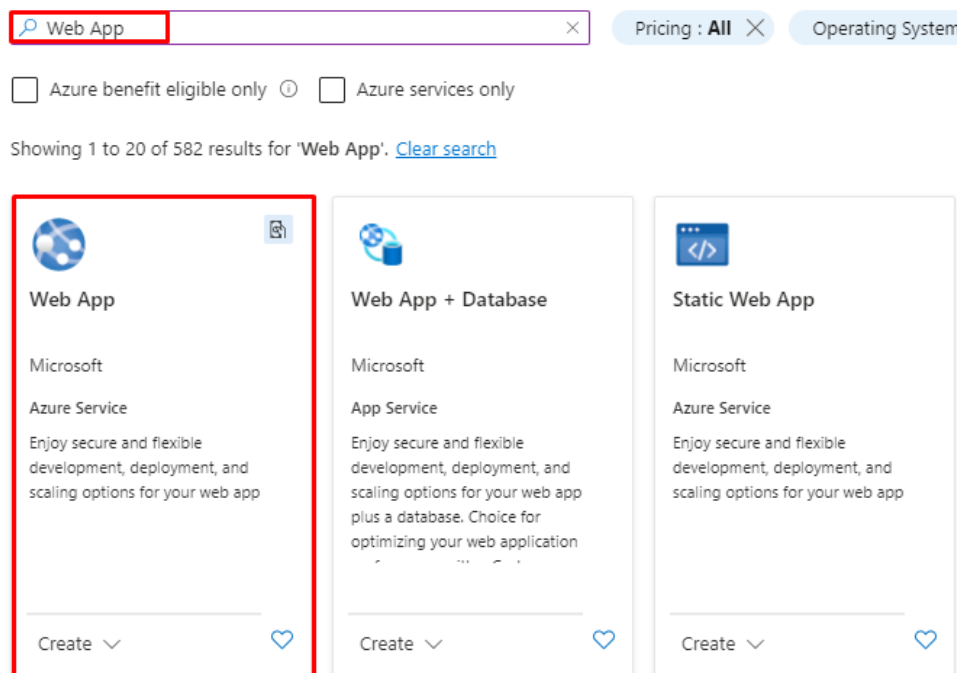
^ Essentials

Resource group (move)	: Azure-App-Service-Lab-KM	Pricing plan	: S1
Status	: Ready	Instance count	: 1
Location	: Poland Central	App(s) / Slots	: 0 / 0
Subscription (move)	: PD - RIP RDC - MSDN - 01	Operating System	: Windows
Subscription ID	: 6168bc6b-a39f-4430-9e73-a36f5d9cd09f	Zone redundant	: Disabled
Tags (edit)	: Add tags		

3. Wdrożenie Azure Web App Service (HTML & FTP)

3.1. Utworzenie Azure Web App Service

- W ramach utworzonej w poprzedniej sekcji Resource Group tworzymy Azure Web App Service



- Name: N/A (ważne aby unikalna globalnie)
- Publish: Code
- Runtime stack: .NET 6
- Operating System: Taki sam jak Azure App Service Plan
- Region: Taki sam jak Azure App Service Plan
- Pricing Plans: Należy wybrać utworzony wcześniej Azure App Service Plan

UWAGA: Jeśli ASP jest niemożliwy do wybrania prawdopodobnie, któryś z parametrów jest niezgodny (Region, Operating System)

- Deployment: N/A
- Networking: N/A (ważne aby Enable public access był wybrany)
- Monitoring:
 - Enable Application Insights: No (monitoring nie jest tematem zajęć, a generowałby koszty i czas potrzebny na utworzenie zasobu)

Zrzut wybranych ustawień znajduje się na kolejnej stronie.

Basics Deployment Networking Monitoring Tags Review + create

Summary



Details

Subscription	6168bc6b-a39f-4430-9e73-a36f5d9cd09f
Resource Group	Azure-App-Service-Lab-KM
Name	agh-lab-km-3
Publish	Code
Runtime stack	.NET 6 (LTS)

App Service Plan

Name	agh-lab-km-asp
Operating System	Windows
Region	Poland Central
SKU	Standard
Size	Small
ACU	100 total ACU
Memory	1.75 GB memory

Monitoring

Application Insights	Not enabled
----------------------	-------------

Deployment

Continuous deployment	Not enabled / Set up after app creation
-----------------------	---

- Weryfikacja:

- Po wciśnięciu na URL powinniśmy zostać przekierowani do działającej witryny:



Your web app is running and waiting for your content

Your web app is live, but we don't have your content yet. If you've already deployed, it could take up to 5 minutes for your content to show up, so come back soon.



</> Supporting Node.js, Java, .NET and more

Haven't deployed yet?
Use the deployment center to publish code or set up continuous deployment.

[Deployment center](#)

Starting a new web site?
Follow our Quickstart guide to get a web app ready quickly.

[Quickstart](#)

- App Service Plan powinien zawierać utworzony App Service:

agh-lab-km-asp | Apps

App Service plan

Search

Filter by name 0 selected

Name	Type	Kind	Resource Group	Status
agh-lab-km-3	App	app	Azure-App-Service-Lab-KM	Running

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Events (preview)

Settings

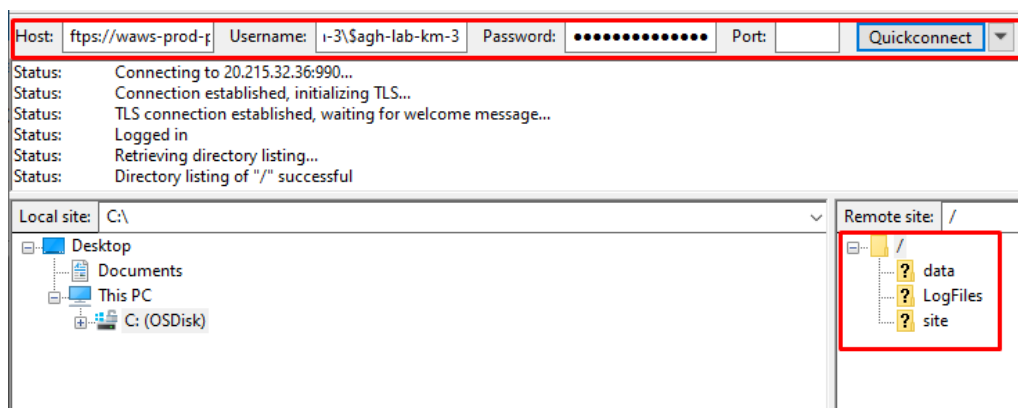
Apps

File system storage

3.2. Wdrożenie pliku HTML

- Należy utworzyć plik index.html (można skorzystać z [Simple HTML](#))
- Następnie przechodzimy do utworzonego Azure App Service
 - Deployment -> Deployment Center -> FTP credentials
- Uruchamiamy program FileZilla
 - Należy połączyć się z serwerem FTP przy pomocy podanych danych autentykacyjnych

UWAGA: Niekiedy mogą pojawić się problemy z połączeniem, należy pamiętać, że pole **Port** powinno pozostać puste.



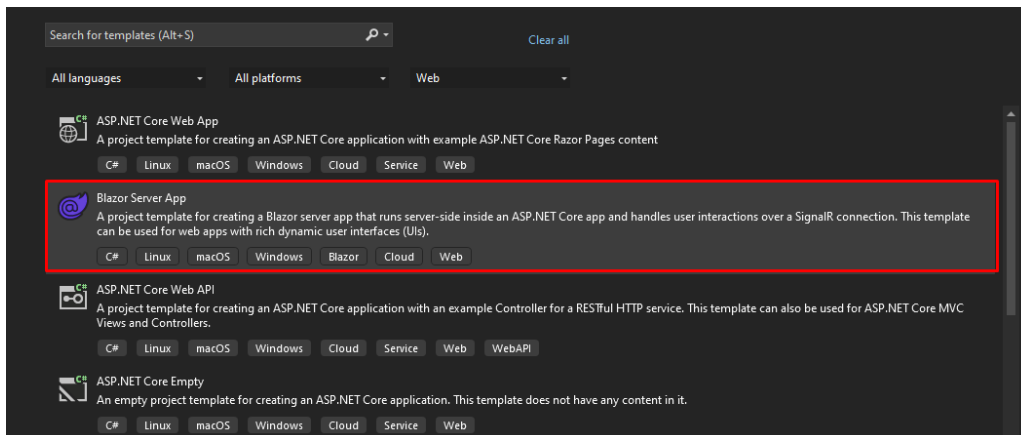
- Kolejnym krokiem jest wgranie pliku index.html do folderu **site/wwwroot**
- Po odświeżeniu/otworzeniu URL naszej strony powinniśmy otrzymać rezultat zgodny z zawartością pliku index.html



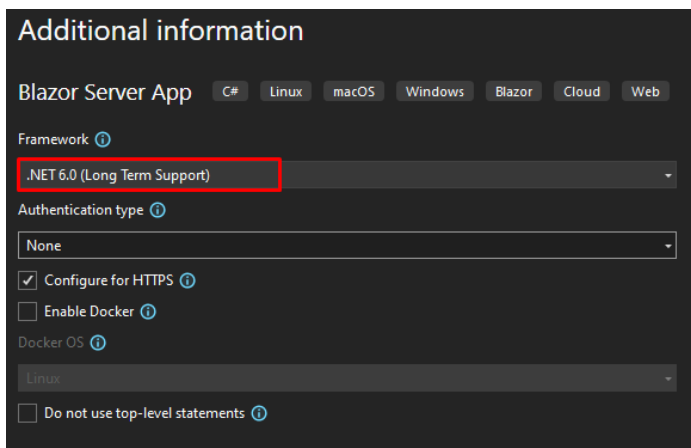
This is an example paragraph. Anything in the **body** tag will appear on the page, just like this **p** tag and its contents.

4. Wdrożenie Azure Web App Service (.NET + Visual Studio 2022)

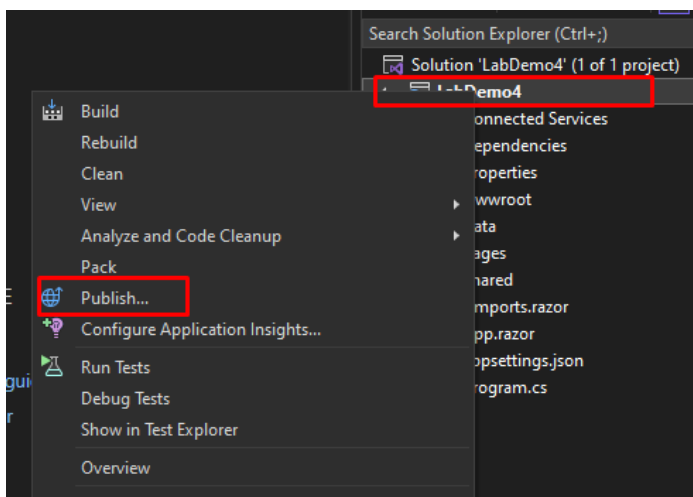
- Tworzymy nowy Azure App Service, tożsamy do tego utworzonego w poprzednim ćwiczeniu
- Uruchamiamy Visual Studio 2022 i tworzymy nowy projekt. Na potrzeby ćwiczeń wybierzemy **Blazor Server App** (aplikacja zawiera w sobie wbudowaną warstwę UI)



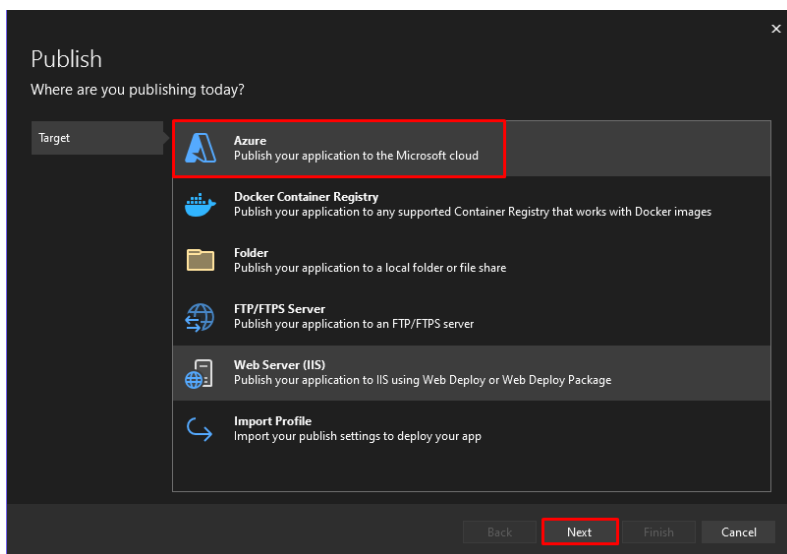
- Przy konfiguracji wybieramy **.NET 6.0** (ważne żeby wersja zgadzała się ze środowiskiem uruchomieniowym wybranym przy tworzeniu App Service w Azure)



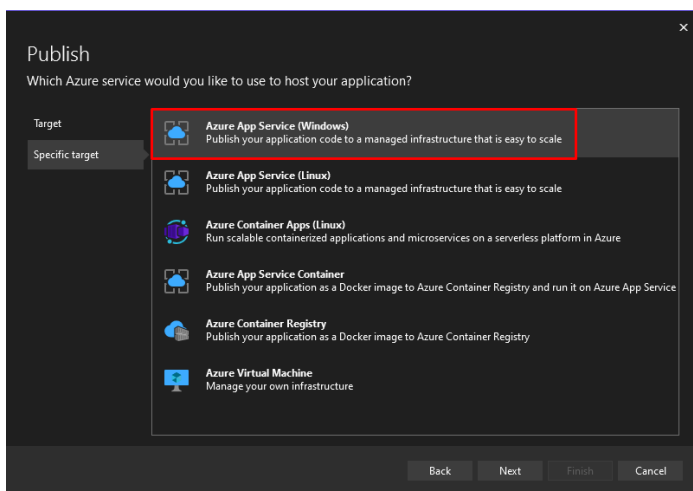
- Dla testów możemy uruchomić aplikację lokalnie, aby zobaczyć jaki jest stan oczekiwany po wdrożeniu
- W drzewie solucji należy wcisnąć PPM na projekt i wybrać opcję **Publish**



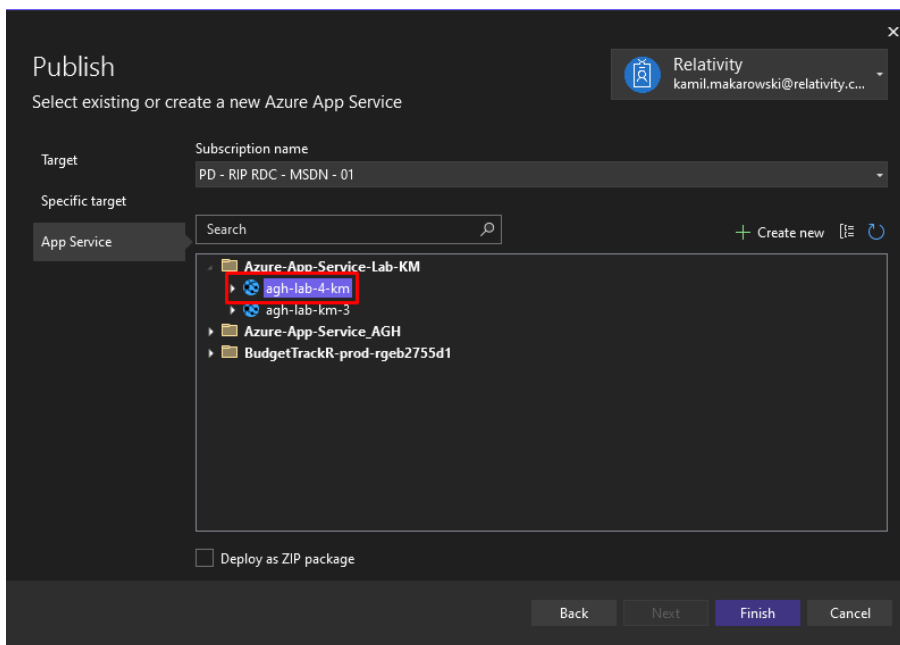
- Kolejnym krokiem jest wybranie miejsca w które chcemy wdrożyć nasz kod. W naszym przypadku jest to Azure App Service więc wybieramy **Azure**



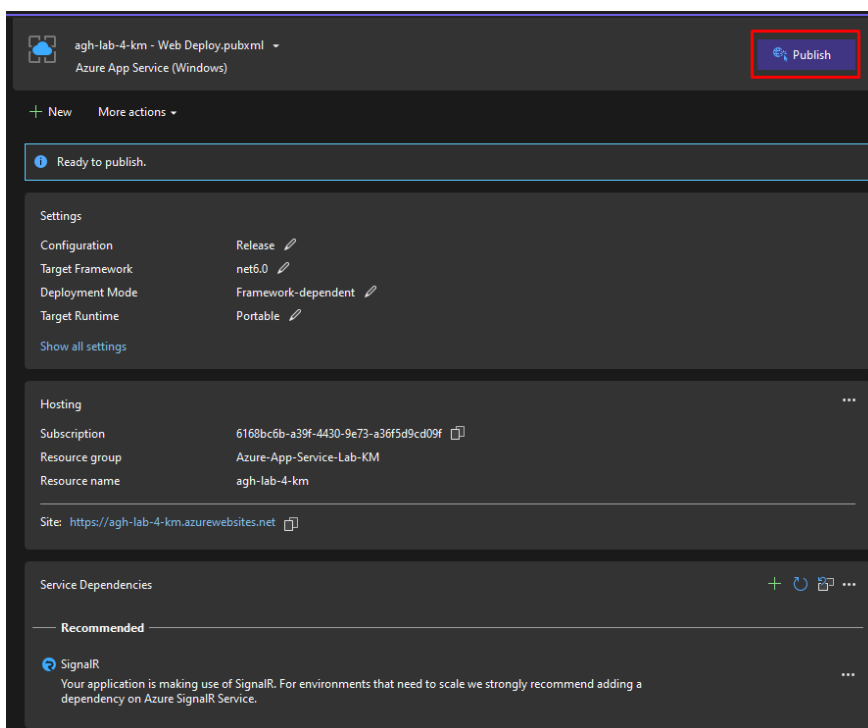
- Następnie wybieramy na jakiego typu Azure App Service chcemy wykonać wdrożenie. Dostępność będzie zależęć od Systemu Operacyjnego, który wybraliśmy przy tworzeniu Azure App Service:



- Następnie wybieramy instancję, na którą chcemy wykonać wdrożenie:



- Rezultatem wykonanej konfiguracji jest „Publish Profile”, który zawiera metadane dotyczące wdrożenia
- Wciskamy **Publish**:

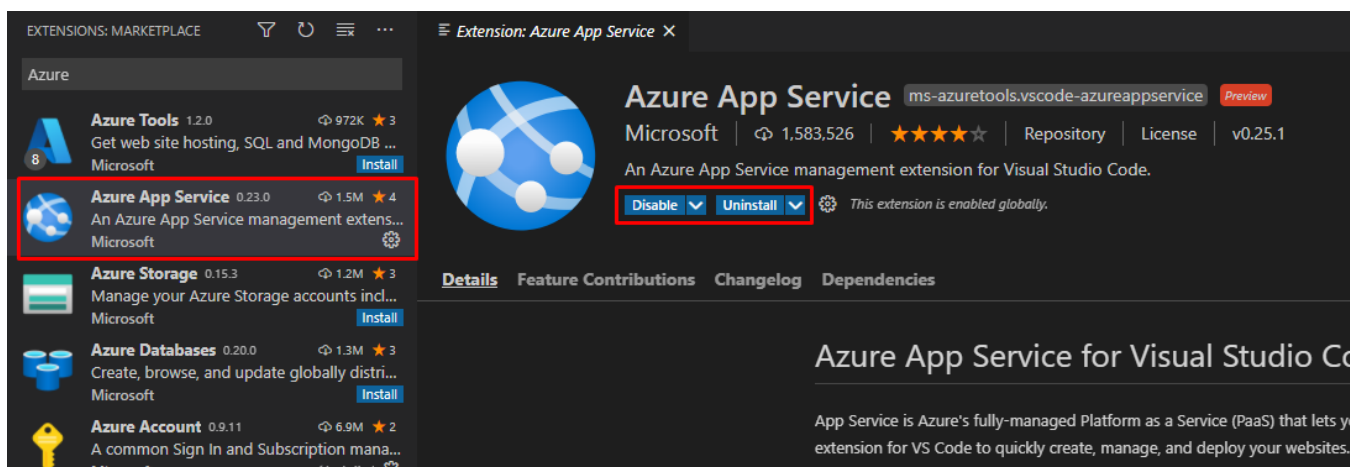


- Na konsoli poniżej możemy monitorować postęp
- Po zakończonym wdrożeniu strona powinna otworzyć się automatycznie

UWAGA: W przypadku .NET 6 można napotkać błąd: **ANCM Failed to Find Native Dependencies**, należy jeszcze raz uruchomić wdrożenie aplikacji klikając **Publish**

5. Wdrożenie Azure Web App Service (React + Visual Studio Code)

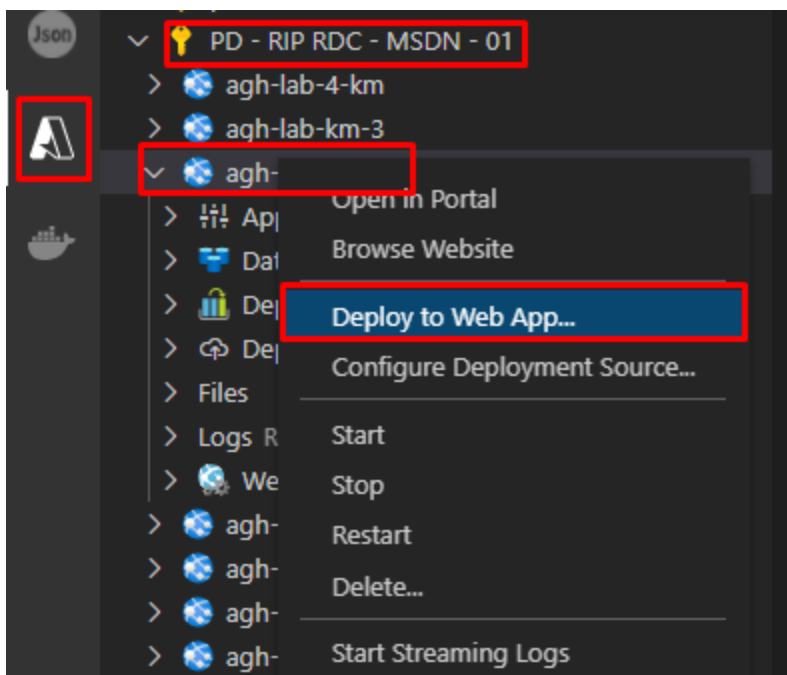
- Należy utworzyć nowy App Service:
 - Stack runtime: Node 18 LTS
- Uruchamiamy Visual Studio Code i tworzymy nową aplikację react korzystając z [React Getting Started](#)
 - `npx create-react-app <nazwa>`
 - `cd <nazwa>`
 - `npm start`
 - `npm run build`
- W Visual Studio Code instalujemy rozszerzenie **Azure App Service**:



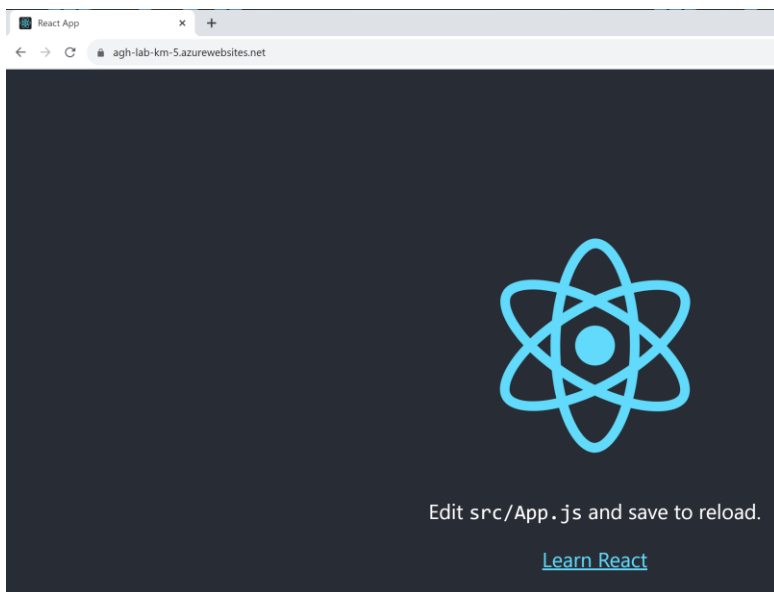
- Musimy też zainstalować rozszerzenie **Azure Account**, w wersji **0.9.0**

UWAGA: Wersja najnowsza 0.9.11 ma błąd, który uniemożliwia zalogowanie się do Azure z poziomu VSC

- Następnie przechodzimy do zakładki **Azure** w lewym panelu i wybieramy odpowiednią subskrypcję oraz App Service, na który chcemy wykonać wdrożenie:



- Następnie z panelu wybieramy folder, który chcemy wdrożyć. W naszym przypadku jest to **build**
- Postęp możemy obserwować w terminalu.
- Po zakończeniu możemy otworzyć aplikację:



UWAGA: To samo można wykonać wciskając PPM bezpośrednio na folder **build** w drzewie plików w VSC i wybierając opcję „**Deploy to web app...**”. Dla celów edukacyjnych można wypróbować też tę drugą metodę.

6. Komunikacja między UI & API - CORS & AppSettings

- Należy ściągnąć kod z [Sample App](#).
- Następnie spróbować go uruchomić lokalnie i zobaczyć czy komunikacja pomiędzy API i UI działa poprawnie
 - Aby uruchomić dwa projekty w VS należy wcisnąć PPM na solucję i wybrać „Set Startup Projects...”
- Komunikacja odbywa się na podstawie ustawienia „API” w pliku Appsettings.Development.json
- Następnie należy wdrożyć obie aplikacje API i UI na Azure zgodnie z ćwiczeniem 4.
- Po uruchomieniu UI i przejściu na zakładkę „Fetch data” nie powinniśmy otrzymywać danych, ponieważ API nie jest połączone z aplikacją.
- Kolejnym krokiem jest zdefiniowanie ustawienia „API”, które przykryje to zdefiniowane w kodzie:

The screenshot shows the Azure portal interface for configuring an application. The left sidebar has a 'Configuration' tab highlighted. The main area shows 'Application settings' with a table containing one entry: 'API' with a hidden value and source 'App Service'. Below this is the 'Connection strings' section, which is currently empty.

Name	Value	Source	Deployment slot setting	Delete	Edit
API	Hidden value. Click to show value	App Service			

Name	Value	Source	Type	Deploym...	Delete	Edit
(no connection strings to display)						

- Od tego momentu komunikacja może już zacząć działać, wynika to z konfiguracji Azure i tego, że aplikacje w domenie *azurewebsites.net* uznawane są za bezpieczne.
- Dla pewności i przeciwiczenia skonfigurujemy również CORS, które zdefiniują, że aplikacja z kodem UI jest bezpieczna dla wdrożonego API:

Home > agh-lab-sampleapp-api

agh-lab-sampleapp-api | CORS

Web App

Search Save Discard

- WebJobs
- MySQL In App
- Service Connector
- Properties
- Locks
- App Service plan
 - App Service plan
 - Quotas
 - Change App Service plan
- Development Tools
 - Clone App
 - Console
 - Advanced Tools
 - App Service Editor (Preview)
 - Extensions
- API
 - API Management
 - API definition
 - CORS**
- Monitoring
 - Alerts

CORS

Cross-Origin Resource Sharing (CORS) allows JavaScript code running in a browser on an external host to interact with all, use "*" and remove all other origins from the list. Slashes are not allowed as part of domain or after TLD. [Learn more](#)

Request Credentials

☐ Enable Access-Control-Allow-Credentials ⓘ

Allowed Origins

<https://agh-lab-sampleapp.azurewebsites.net>

- Ostatnim krokiem jest weryfikacja, czy jesteśmy w stanie odczytać temperaturę z naszego API.
 - Dla potwierdzenia można zmodyfikować kod w *WeatherForecastController.cs*

```
public class WeatherForecastController : ControllerBase
{
    private static readonly string[] Summaries = new[]
    {
        "Freezing & Hot"
    };
    ...
}
```

- Następnie ponownie wdrożyć API i sprawdzić czy wszystkie odczytane dane zwracają „Freezing & Hot”

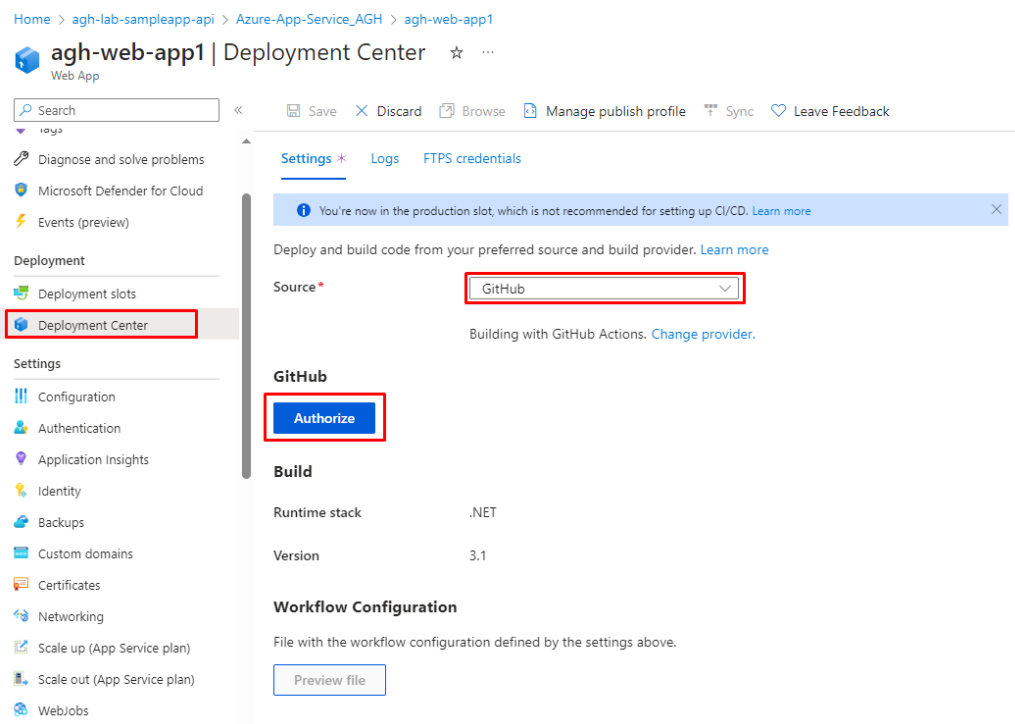
7. Podstawy CI/CD

7.1. Deployment Center

- W GitHub tworzymy publiczne repozytorium
 - W przypadku .gitignore warto wybrać szablon **Visual Studio**, co ograniczy liczbę plików
- Następnie klonujemy repozytorium na lokalny dysk
- W folderze tworzymy nowy projekt w Visual Studio jako **Blazor Server App**

UWAGA: plik *.sln powinny znajdować się w root repozytorium

- Następnie wypychamy zmiany na zdalny branch **main**. Przed push upewnijmy się, że projekt działa i buduje się lokalnie
- Następnie w App Service (nowym/istniejącym) przechodzimy do zakładki **Deployment Center**:
 - Wybieramy GitHub i klikamy Authorize



- Następnie wybieramy **Change Provider** i **App Service Build Service** co pozwoli przenieść odpowiedzialność budowania na Azure

i You're now in the production slot, which is not recommended for setting up CI/CD. [Learn more](#)

Deploy and build code from your preferred source and build provider. [Learn more](#)

Source *

Building with App Service Build Service. [Change provider.](#)

GitHub

App Service will place a webhook on the chosen repository. When a new commit is pushed to the repository, App Service will pull your code, build your application, and deploy it to your web application. If you are using a private repository, you may need to enable additional permissions on GitHub. [Learn more](#)

Signed in as kamil-makarowski-relativity [Change Account](#)

Organization *

Repository *

Branch *

Build provider

☐ GitHub Actions

☒ App Service Build Service

☐ Azure Pipelines

Ok

Cancel

- Wybieramy z listy repozytorium oraz branch i wciskamy **Save**
- Następnie możemy zmienić kod w pliku **WeatherForecastService.cs** na:

```
private static readonly string[] Summaries = new[]
{
    "Freezing in CI/CD"
};
```

- Następnie wypychamy zmiany na zdalne repozytorium i patrzymy na logi w Deployment Center:

[Save](#) [Discard](#) [Browse](#) [Manage publish profile](#) [Sync](#) [Leave Feedback](#)

Settings Logs FTPS credentials

[Refresh](#) [Delete](#)

Time	Commit ID	Commit Author	Status	Message
Thursday, November 2, 2023 (2)				
11:43:33 AM +01:00	51f8485	Kamil Makarowski	Success (Active)	freezing in CI/CD
11:41:59 AM +01:00	1cd56d9	Kamil Makarowski	Success	initial commit

- Aplikacja powinna wyświetlać:







Weather forecast

This component demonstrates fetching data from a service.



Date	Temp. (C)	Temp. (F)	Summary
11/3/2023	-18	0	Freezing in CI/CD
11/4/2023	51	123	Freezing in CI/CD
11/5/2023	43	109	Freezing in CI/CD
11/6/2023	4	39	Freezing in CI/CD
11/7/2023	-2	29	Freezing in CI/CD

7.2. Wdrożenie z testami

- Tworzymy nowy Azure Web App
- Konfigurujemy go zgodnie z zadaniem 7.1, przy czym tym razem wybieramy **GitHub Actions**

 Save  Discard  Browse  Manage publish profile  Sync  Leave Feedback

[Settings](#) * [Logs](#) [FTPS credentials](#)

 You're now in the production slot, which is not recommended for setting up CI/CD. [Learn more](#) 

Deploy and build code from your preferred source and build provider. [Learn more](#)


Source *

GitHub

Building with GitHub Actions. [Change provider.](#)

GitHub

App Service will place a GitHub Actions workflow in your chosen repository to build and deploy your app whenever there is a commit on the chosen branch. If you can't find an organization or repository, you may need to enable additional permissions on GitHub. You must have write access to your chosen GitHub repository to deploy with GitHub Actions. [Learn more](#)

Signed in as kamil-makarowski-relativity [Change Account](#) 

Organization *

kamil-makarowski-relativity

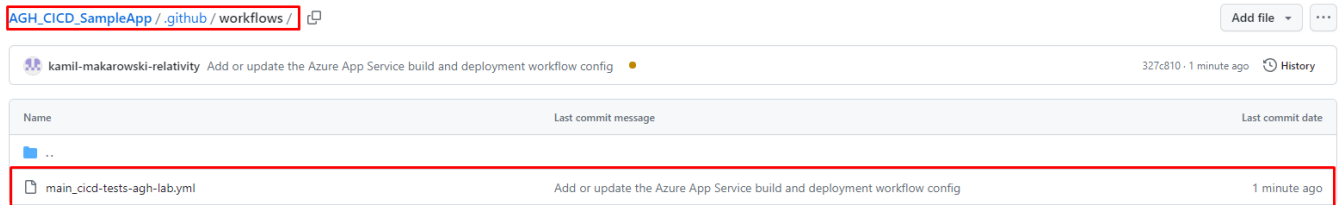
Repository *

AGH_CICD_SampleApp

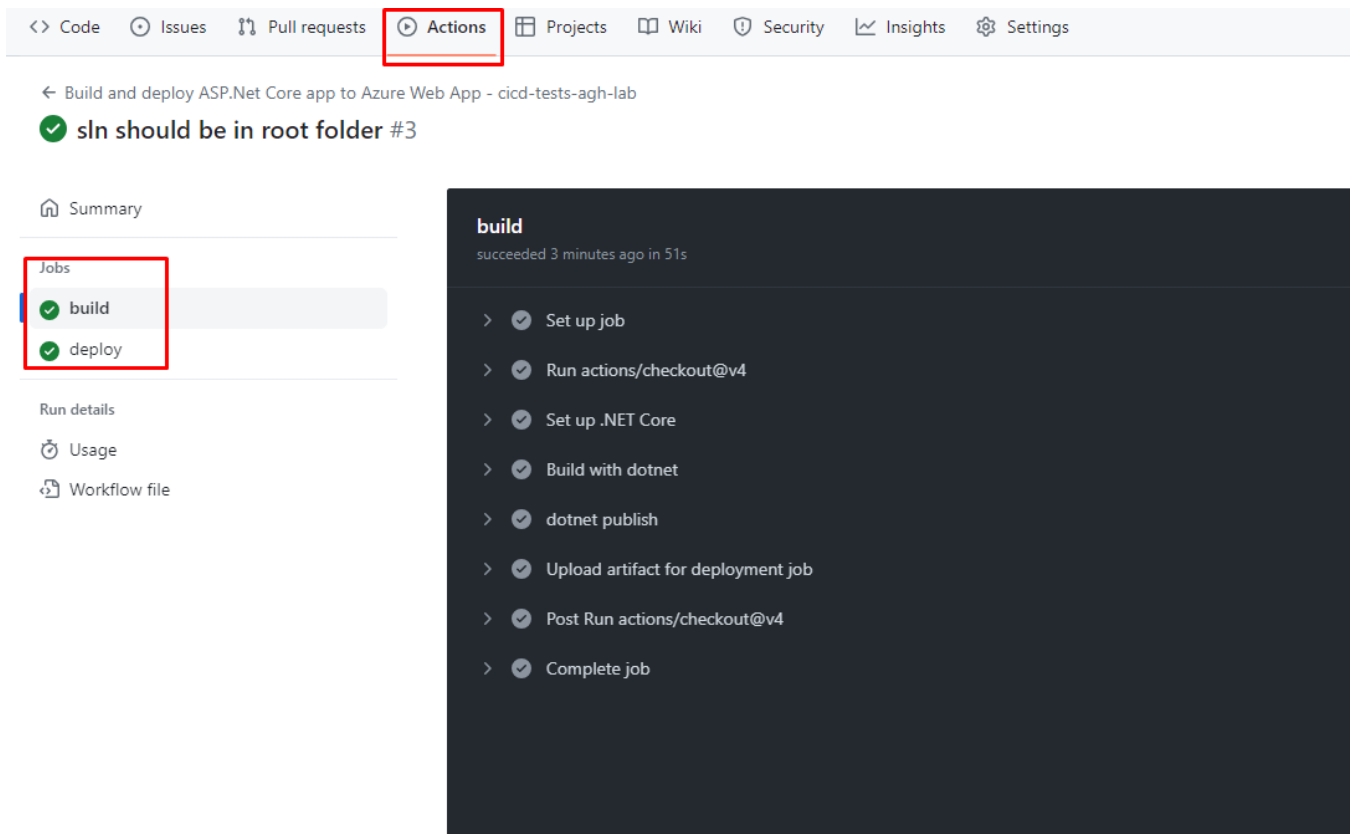
Branch *

main

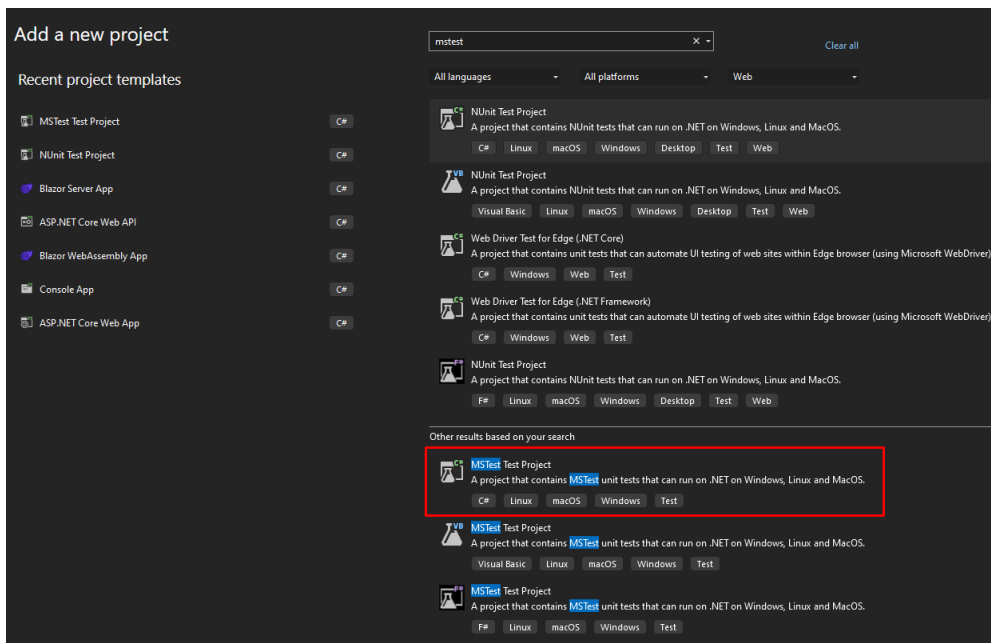
- Możemy również podejrzeć plik z workflow, który zostanie utworzony i będzie wykorzystywany przez system CI/CD do wdrożenia wciskając **Preview file**
- Po wciśnięciu **Save** możemy zobaczyć, że Azure dodał do naszego repozytorium nowy plik:



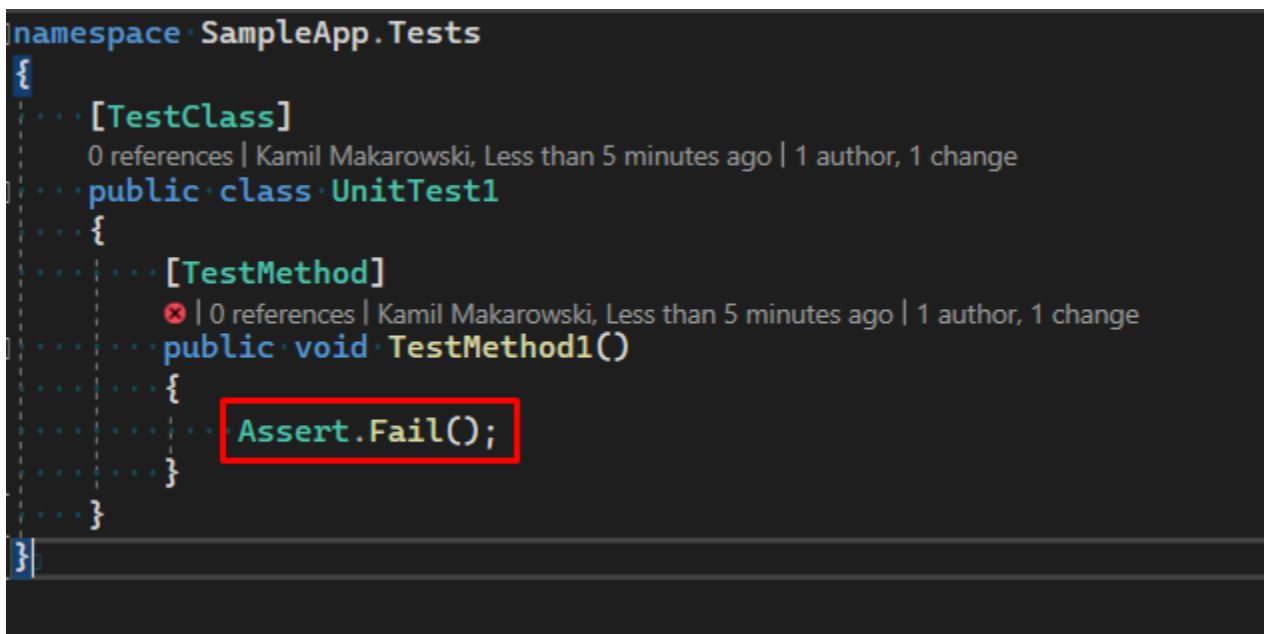
- Build automatycznie powinien się uruchomić i wykonać, jego postęp można śledzić w Azure albo w GitHub w zakładce **Actions**:



- Kolejnym krokiem jest dodanie testów do naszego rozwiązania. W tym celu na solucji klikamy PPM i wybieramy Add -> New Project:



- Zgodnie z dobrymi praktykami można go nazwać *<NazwaProjektu>.Tests*
- W klasie **UnitTests1.cs** dodajemy **Assert.Fail()**



- Test po uruchomieniu powinien zapalić się na czerwono
- Następnie wypychamy zmiany do zdalnego repozytorium
 - Build ponownie przejdzie na zielono mimo failującego testu, dzieje się tak dlatego, że nie skonfigurowaliśmy uruchamiania testów w naszym pipeline
- W celu skonfigurowania kroku testowego należy zmodyfikować plik *.yml w folderze .github/workflows
 - Można to zrobić bezpośrednio w GitHub lub lokalnie
 - Należy pomiędzy krokiem **Build with dotnet**, a **dotnet publish** dodać:

- name: Run tests
run: dotnet test

UWAGA: Wcięcia są ważne

- Po wrzuceniu zmian do zdalnego repozytorium build powinien zakończyć się błędem związanym z błędnymi testami
- Błędny build powinien również być widoczny w Azure w zakładce Logs:

Save Discard Browse Manage publish profile Sync Leave Feedback

Settings Logs FTPS credentials

Refresh Delete


Time	Commit ID	Logs	Commit Author	Status	Message
Thursday, November 2, 2023 (17)					
11/2 2023, 1:56:53 PM +01:00	e60d039	Build/Deploy Logs	kamil-makarowski-relativity	Failed	Update main_cicd-tests-agh-lab.yml
11/2 2023, 1:50:49 PM +01:00	74cba73	Build/Deploy Logs	Kamil Makarowski	Failed	change project typ eto MSTest
11/2 2023, 1:27:04 PM +01:00	b338a28	App Logs	N/A	Success (Active)	Update main_cicd-tests-agh-lab.yml
11/2 2023, 1:25:28 PM +01:00	a44fea5	Build/Deploy Logs	kamil-makarowski-relativity	Success	Update main_cicd-tests-agh-lab.yml
11/2 2023, 1:23:03 PM +01:00	84f87de	Build/Deploy Logs	kamil-makarowski-relativity	Failed	Update main_cicd-tests-agh-lab.yml
11/2 2023, 1:19:57 PM +01:00	98b75b0	Build/Deploy Logs	kamil-makarowski-relativity	Failed	Update main_cicd-tests-agh-lab.yml

- W celu naprawienia testów należy zmodyfikować kod na **Assert.Pass()**

8. Deployment Slots

- Nawigujemy do Web App stworzonego w zadaniu 3
- Wchodzimy w Deployment Slots, a następnie tworzymy nowy. Możemy go nazwać „test”.

Save Discard Add Slot Swap Logs Refresh

 Deployment Slots

Deployment slots are live apps with their own hostnames. App content and configurations elements can be swapped between two deployment slots, including the production slot.

NAME	STATUS	APP SERVICE PLAN	TRAFFIC %
agh-lab-km-3 PRODUCTION	Running	agh-lab-km-asp	100
agh-lab-km-3-test	Running	agh-lab-km-asp	0

- Nowopowstały slot jest osobnym Web Appem, do którego możemy się odwoływać tak jak do innych.
- Wchodzimy do `<Nazwa_Web_App>-test`, a następnie przechodzimy do *Deployment Center* -> *FTPS Credentials*.
- Powtarzamy kroki z zadanie 3.2 i wdramy plik index.html, przy czym wcześniej należy go zmodyfikować na cokolwiek innego, aby komunikat był inny. Dzięki temu będziemy wiedzieć, który z nich został wyświetlony
- Następnie w sekcji Deployment Slots **głównego Web App** ustawiamy dla naszego Slotu „test” Traffic na poziomie **50%**

- Następnie wracamy do strony głównej i Web App i uruchamiamy główny URL z sekcji **Default Domain**, aby wyświetlić zawartość strony.

[Browse](#)
[Stop](#)
[Swap](#)
[Restart](#)
[Delete](#)
[Refresh](#)
[Download publish profile](#)
[Reset publish profile](#)
[Share to mobile](#)
[Send us your feedback](#)

[Click here to access Application Insights for monitoring and profiling for your app.](#)

[Essentials](#)

Resource group [\(move\)](#) : [Azure-App-Service-Lab-KM](#)

Status : Running

Location [\(move\)](#) : Poland Central

Subscription [\(move\)](#) : [PD - RIP RDC - MSDN - 01](#)

Subscription ID : 6168bc6b-a39f-4430-9e73-a36f5d9cd09f

Tags [\(edit\)](#) : [Add tags](#)

Default domain : [agh-lab-km-3.azurewebsites.net](#)


App Service Plan : [agh-lab-km-asp \(\\$1: 1\)](#)

Operating System : Windows

Health Check : Not Configured

- Kolejnym krokiem jest przeprowadzenie testów. Otwieramy nowe okno w trybie incognito i próbujemy uruchomić stronę kopiując adres. Powtarzamy operację kilka razy i sprawdzamy czy naprzemiennie wyświetla nam się wersja „prod” i wersja „test” pomimo tego samego adresu.
- Ostatnim krokiem, który symuluje wdrożenie produkcyjne jest wykonanie operacji **Swap**. Wracamy do Deployments Slots naszego Web App i wybieramy „Swap”:

[Save](#)
[Discard](#)
[Add Slot](#)
[Swap](#)
[Logs](#)
[Refresh](#)


Deployment Slots

Deployment slots are live apps with their own hostnames. App content and configurations elements can be swapped between two deployment slots, including the production slot.

NAME	STATUS	APP SERVICE PLAN	TRAFFIC %
agh-lab-km-3 PRODUCTION	Running	agh-lab-km-asp	50
agh-lab-km-3-test	Running	agh-lab-km-asp	50

- Wybieramy Source i Target, a następnie wciskamy „Swap”

Swap

● Source
 agh-lab-km-3-test

● Target **PRODUCTION**
 agh-lab-km-3

ⓘ Swap with preview can only be used with sites that have deployment slot settings enabled

☐ Perform swap with preview

Config Changes

This is a summary of the final set of configuration changes on the source and target deployment slots after the swap has completed.

● Source Changes		● Target Changes	
SETTING	TYPE	OLD VALUE	NEW VALUE
No Changes			

- Od teraz każde wywołanie default domain będzie skutkować wyświetleniem wersji pochodzącej z `<Nazwa_Web_App>-test`.

9. OBOWIĄZKOWE NA KONIEC ZAJĘĆ

- Na koniec zajęć zalecane jest usunięcie całej utworzonej **Resource Groupy**
- Jeśli natomiast bardzo chcecie zachować rezultat dzisiejszych zajęć do analizy można wykonać:
 - Dobrą opcją jest usunięcie slotu staging, wyłączenie Always on w WebAppie (configuration -> General settings -> Always On na false) i zmiana planu na F1 (app service plan -> Scale up)
 - Wyłączenie wszystkich utworzonych Web App:

