



Hands-on Lab

Azure DevOps Hands-on Lab Basic

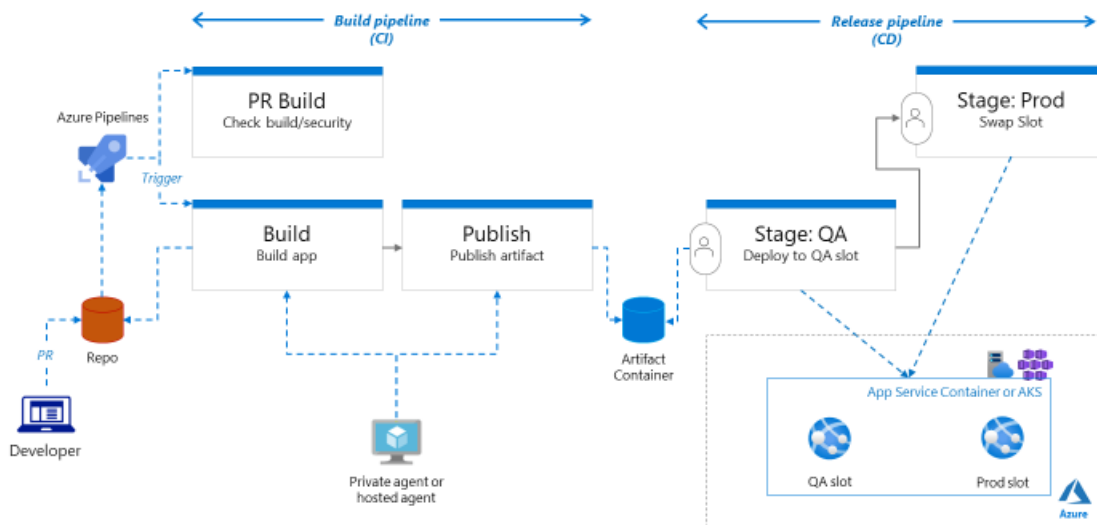
Table of Contents

1	Introduction	3
2	Lab0: DevOps setup.....	4
	2.1 Config private agent	4
	2.2 Create a new project.....	4
3	Lab2: Build pipeline (Classic).....	7
	3.1 Setup Repo.....	7
	3.2 Config build pipeline	7
	3.3 Queue build	9
4	Lab2: Build pipeline (YAML)	11
	4.1 Edit build pipeline.....	11
	4.2 Setup pipeline trigger	13
5	Lab3: Release pipeline (UI)	14
	5.1 Setup target VM(Web App)	14
	5.2 Config release pipeline	15
	5.3 Config QA stage job	15
	5.4 Config Prod stage job	17
	5.5 Trigger release	18
	5.6 Test deployment	18
6	Appendix.....	19
	6.1 Git clone/Credential.....	19
	6.2 Service Connection	19

1 Introduction

본 hands-on lab 에서는 .NET 개발자를 위한 DevOps CI/CD pipeline 실습을 진행합니다.

DevOps CI/CD pipeline



실습 내용:

Lab	Description	Note
0	DevOps Setup	
1	Build pipeline (Classic)	
2	Build pipeline (YAML)	
3	Release pipeline (UI)	

2 Lab0: DevOps setup

사전준비 사항: Azure 구독 (구독 Owner 권한 권장)과 Azure DevOps 계정이 필요.

Azure DevOps lab 을 진행하기에 앞서 필요한 리소스들을 미리 준비합니다.

- 신규 DevOps 신규 Project 생성
- Private agent 생성 (선택사항)
 - Agent pool 생성 (예: *winpool*)
 - PAT 토큰 생성 ([agent 권한](#))
 - Windows OS 또는 Linux VM (Ds4_v3 권장) 생성
 - 개발도구 설치
 - Agent VM 에 개발 관련 도구 설치: .NET SDK 등
 - Agent 설치 (상세 내용은 2.2 참조)

2.1 Create a new project

[New project]를 클릭하여 신규 프로젝트를 생성합니다.

Create new project

×

Project name *

devops-lab

✓

Description

Azure DevOps HoL

Visibility

Public

Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private

Only people you give access to will be able to view this project.

Advanced

2.2 Config private agent

Private agent 로 사용될 VM 을 생성하고, 개발 관련 도구를 설치합니다. (.NET SDK 는 pipeline 에서 설치 가능)

- .NET SDK: <https://dotnet.microsoft.com/download>

- **팁:** 개발도구가 먼저 설치되어야 agent 가 정상적으로 인식.

Project settings->Agent Pools-> **[Add pool]**을 클릭하고 속성을 입력하여 신규 pool 을 생성합니다.

Add agent pool ✕

Agent pools are shared across an organization.

Pool to link:

☒ New ☐ Existing

Pool type:

Self-hosted ▾

Name:

winpool

Description (optional):

Windows Server 2016+

[Markdown supported.](#)

Pipeline permissions:

☒ Grant access permission to all pipelines

앞서 생성한 pool 을 선택하고 **[New agent]** 클릭합니다. 화면에 표시된 설치 스크립트를 복사 후, Agent VM 에서 실행합니다.

노트: 본 실습은 private agent 없이도 진행이 가능하지만, 향후 enterprise 환경에 좀더 적합한 DevOps 환경 구축을 위해 구성하는 것을 권장.


Get the agent

Windows macOS Linux

x64
x86

System prerequisites

Configure your account
Configure your account by following the steps outlined here.

Download the agent
[Download](#) 

Create the agent

```
PS C:\> mkdir agent ; cd agent
PS C:\agent> Add-Type -AssemblyName System.IO.Compression.FileSystem ;
[System.IO.Compression.ZipFile]::ExtractToDirectory("$HOME\Downloads\vsts-agent-win-x64-2.175.2.zip", "$PWD")
```

Configure the agent [Detailed instructions](#)

```
PS C:\agent> .\config.cmd
```

Optionally run the agent interactively
If you didn't run as a service above:

```
PS C:\agent> .\run.cmd
```

That's it!
[More Information](#)

세부 설치 내용은 아래 문서링크를 참조하시기 바랍니다.

<https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops>

3 Lab2: Build pipeline (Classic)

3.1 Setup Repo

Repos->Files->Import a repository->**[Import]** 선택하고, Github 에 공개된 예제 (<https://github.com/iljoong/devops-basic>) 소스 Repo 를 import 하여 추가합니다.

Import a Git repository ×

Repository type

 Git ▼

Clone URL *

☐ Requires Authentication


3.2 Config build pipeline


Pipelines->Pipelines->**[Create/New Pipeline]**을 선택하여 신규 빌드 파이프라인을 생성합니다.
이때 아래의 화면과 같이 **Use the classic editor** 클릭하여 진행합니다.


Connect **Select** **Configure** **Review**


New pipeline


Where is your code?


 **Azure Repos Git** **YAML**
Free private Git repositories, pull requests, and code search

 **Bitbucket Cloud** **YAML**
Hosted by Atlassian

 **GitHub** **YAML**
Home to the world's largest community of developers

 **GitHub Enterprise Server** **YAML**
The self-hosted version of GitHub Enterprise

 **Other Git**
Any generic Git repository

 **Subversion**
Centralized version control by Apache

Use the classic editor to create a pipeline without YAML.

Source 선택에서 **Azure Repos Git** 을 선택하여 다음으로 이동합니다.

Select a source



Team project

devopstest

Repository

devopstest

Default branch for manual and scheduled builds

main

Continue

Template 선택에서 asp.net core 를 검색하여 선택하여 완료합니다.

Select a template

Or start with an [Empty job](#)

.net core

Configuration as code



YAML

Looking for a better experience to configure your pipelines using YAML files? Try the new YAML pipeline creation experience. [Learn more](#)

Others



ASP.NET Core

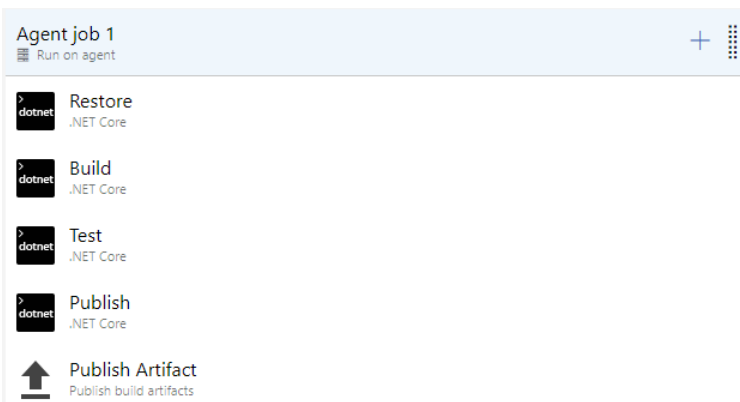
Build and test an ASP.NET Core web application.



ASP.NET Core (.NET Framework)

Build an ASP.NET Core web application that targets the full .NET Framework.

기본적으로 아래와 같이 **restore, build, test, publish, publish artifact** 태스크가 생성됩니다.



Agent pool 은 앞서 생성한 Private agent pool 을 선택하거나 *Microsoft-hosted agent (ubuntu-latest)*를 선택합니다.

Agent job ⓘ

Display name *

Agent job 1

Agent selection ^

Agent pool ⓘ | Pool information | Manage ↗

Azure Pipelines

Agent Specification *

ubuntu-latest

Default 로 생성된 Task 에서 *Test Task*를 삭제하고, Agent Job 의 **[+]**를 클릭하여 **Use .NET core Task**를 추가하고 SDK 버전을 지정합니다.

The screenshot shows the Azure Pipelines 'Agent job 1' configuration page. On the left, under the 'Tasks' tab, a list of tasks is shown for 'Agent job 1'. The task 'Use .NET Core sdk 6.0.x' is selected. On the right, the configuration for this task is displayed. The 'Task version' is set to '2.*'. The 'Display name' is 'Use .NET Core sdk 6.0.x'. The 'Package to install' is 'SDK (contains runtime)'. The 'Use global json' checkbox is unchecked. The 'Version' is set to '6.0.x'. The 'Include Preview Versions' checkbox is unchecked. The 'Advanced' section is collapsed.

3.3 Queue build

Build pipeline 을 저장한 후, 빌드를 실행하고 Agent 의 log 를 확인합니다.

← Jobs in run #20220913.1

dotnetapp-aspnet-msft-ci

Jobs

✓ Agent job 1	33s
✓ Initialize job	5s
✓ Checkout dotnetapp@master to s	2s
✓ Use .NET Core sdk 6.0.x	9s
✓ Restore	4s
✓ Build	7s
✓ Publish	3s
✓ Publish Artifact	1s
✓ Post-job: Checkout dotnetapp@...	<1s
✓ Finalize Job	<1s
✓ Report build status	<1s

✓ Build

```

1 Starting: Build
2 =====
3 Task : .NET Core
4 Description : Build, test, package, or publish a dotnet application, or run a custom dotnet command
5 Version : 2.288.1
6 Author : Microsoft Corporation
7 Help : https://docs.microsoft.com/azure/devops/pipelines/tasks/build/dotnet-core-cli
8 =====
9 Info: .NET Core SDK/runtime 2.2 and 3.0 are now End of Life(EOL) and have been removed from all hosted agents. If you're using these SDK/runtimes on
10 /opt/hostedtoolcache/dotnet/dotnet build /home/vsts/work/1/s/app/dotnetapp.csproj -dl:CentralLogger,"/home/vsts/work/_tasks/DotNetCoreCLI_5541a522-6
11 MSBuild version 17.3.0+92e077650 for .NET
12 Determining projects to restore...
13 Restored /home/vsts/work/1/s/app/dotnetapp.csproj (in 176 ms).
14 dotnetapp -> /home/vsts/work/1/s/bin/Release/net6.0/dotnetapp.dll
15
16 Build succeeded.
17 0 Warning(s)
18 0 Error(s)
19
20 Time Elapsed 00:00:06.26
21 Info: Azure Pipelines hosted agents have been updated and now contain .Net 5.x SDK/Runtime along with the older .Net Core version which are currentl
22 Finishing: Build

```

빌드가 정상적으로 완료되면 최종 Build Artifact 는 container 의 drop 폴더로 퍼블리쉬됩니다.

Manually run by Il Joong Kim

Repository and version

dotnetapp
 master 839082d8

Time started and elapsed

Just now
 42s

Related

0 work items
 1 published: 1 consumed

노트: Artifact 는 Azure DevOps 에서 제공되는 클라우드 내의 shared infrastructure 에 저장되며, private infrastructure 에 저장을 원할 경우 [사용자 지정 file share 에 저장](#)할 수 있음.

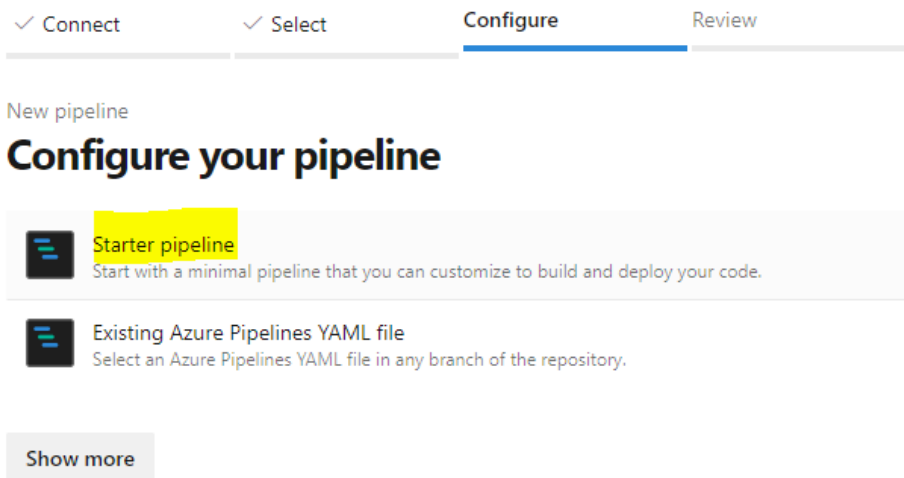
4 Lab2: Build pipeline (YAML)

GUI 기반인 Classic 방식으로 pipeline 을 손쉽게 작성할 수 있지만, 이방식을 사용하면 pipeline 을 개발 소스와 함께 관리할 수 없습니다. 프로젝트를 이전하거나 참조할 때 재사용할 수 없는 단점이 있습니다.

새로운 YAML 방식은 개발 소스와 함께 pipeline 을 코드화 하여 관리할 수 있습니다. 자세한 Azure DevOps YAML 구조 및 문서는 링크(<https://aka.ms/yaml>)를 참조하시기 바랍니다.

4.1 Edit build pipeline

신규 pipeline 을 생성하고, Starter pipeline 템플릿을 선택합니다.



기본 내용은 삭제하고, 앞서 생성한 build pipeline 을 아래와 같은 YAML 형식으로 작성합니다. 화면 우측 **assistant** 의 기능을 이용하여 필요한 Task 를 검색하여 템플릿으로 추가할 수 있습니다.

팁: 빌드 명은 default 로 project 이름으로 생성되어 메뉴에서 원하는 이름으로 수정.

```

# Starter pipeline
trigger:
#- master
- none

pool:
  vmImage: ubuntu-latest

variables:
- name: buildConfiguration
  value: 'Release'

steps:
- task: UseDotNet@2
  displayName: 'Use .NET Core sdk 6.0'
  inputs:
    version: 6.0.x
- task: DotNetCoreCLI@2
  inputs:
    command: 'restore'
    projects: '**/*.csproj'
- task: DotNetCoreCLI@2
  displayName: Build
  inputs:
    projects: '**/*.csproj'
    arguments: '--configuration $(buildConfiguration)'
- task: DotNetCoreCLI@2
  displayName: Publish
  inputs:
    command: publish
    publishWebProjects: False
    projects: '**/*.csproj'
    arguments: '--configuration $(buildConfiguration) --
output $(build.artifactstagingdirectory)'
    zipAfterPublish: True
- task: PublishBuildArtifacts@1
  displayName: 'Publish Artifact'
  inputs:
    PathtoPublish: '$(build.artifactstagingdirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'
  condition: succeededOrFailed()

```

팁: Classic Editor 에서 해당 Task 를 YAML import 해서 추가할 수 있음.

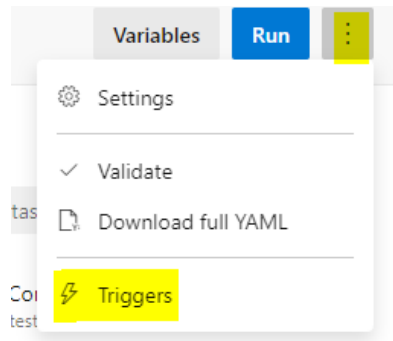
Pipeline 작성이 완료되면 앞서 실행했던 Classic pipeline 과 동일하게 빌드를 실행하여 정상적으로 실행되는지 확인합니다.

4.2 Setup pipeline trigger

소스가 commit 되었을 때 빌드가 실행되는 Continuous Integration (CI)를 구성하고자 할 때 YAML 내에서 아래와 같이 trigger 를 편집합니다.

```
trigger:
- master
```

또는, YAML 편집창 상단 우측 메뉴를 통해 추가 Trigger 메뉴를 선택하여 설정할 수 있습니다.



Class editor 와 동일한 UI 로 Triggers 탭에서 상세 설정이 가능합니다.

☒ Override the YAML continuous integration trigger from here

☐ Disable continuous integration

☒ Enable continuous integration

☐ Batch changes while a build is in progress

Branch filters

Type

Include

Branch specification

master

+ Add

Path filters

+ Add

5 Lab3: Release pipeline (UI)

5.1 Setup target VM(Web App)

배포 타겟으로 Web App 을 생성합니다. 기본 사이트(Production slot)와 QA Slot 을 미리 생성합니다.

Create Web App ...

Basics Deployment Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Microsoft Azure Internal Consumption (Main) ▼

Resource Group * ⓘ

test-vmss ▼

[Create new](#)

Instance Details

Need a database? [Try the new Web + Database experience.](#)

Name *

Web App name.

.azurewebsites.net

Publish *

☒ Code ☐ Docker Container ☐ Static Web App

Runtime stack *

.NET 6 (LTS) ▼

Operating System *

☐ Linux ☒ Windows

Region *

Korea Central ▼

ⓘ Not finding your App Service Plan? Try a different region or select your App Service Environment.

App Service Plan

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more](#)

Windows Plan (Korea Central) * ⓘ

ikappplan (S1) ▼

[Create new](#)

Sku and size *

Standard S1

100 total ACU, 1.75 GB memory

* QA slot 을 생성하기 위해서는 Standard 이상의 SKU 생성 필요

5.2 Config release pipeline

신규 릴리즈 파이프라인을 생성하고, 아래와 같이 파이프라인을 구성합니다.

The screenshot displays the Azure DevOps Release Pipeline configuration page. The top navigation bar includes 'All pipelines > Release-pipeline-bg'. Below this, there are tabs for 'Pipeline', 'Tasks', 'Variables', 'Retention', 'Options', and 'History'. The main area is divided into 'Artifacts' and 'Stages'. The 'Artifacts' section shows a build artifact named '_dotnetapp_aspnet_ci' with a 'Schedule not set' icon. The 'Stages' section shows two stages: 'Deploy to QA' (1 job, 1 task) and 'Swap QA/Prod' (1 job, 1 task). On the right, the 'Artifact' details panel is open, showing the build name 'Build - _dotnetapp_aspnet_ci', project 'dotnetapp', source 'dotnetapp-aspnet-msft-yaml-ci', default version 'Latest', and source alias '_dotnetapp_aspnet_ci'.

Artifacts: Build (project) – *buildid* 를 가져오기 위해 필요

Stages: QA 와 Prod 용 두개의 stage 생성

주의: Artifact 의 source alias 명에 hyphen('-') 이 들어가지 않게 수정. 예 `_demo-build-ci` 는 `_demo_build` 로 수정 (Linux 에서 환경변수가 정상적으로 설정 안됨)

릴리즈 파이프라인에서 사용될 변수를 *Variables* 편집창에 추가합니다.






The screenshot shows the 'Variables' tab in the Azure DevOps Release Pipeline configuration page. The left sidebar has tabs for 'Pipeline variables', 'Variable groups', and 'Predefined variables'. The main area has a search bar 'Filter by keywords' and a table with columns 'Name' and 'Value'. At the bottom, there is a '+ Add' button.

5.3 Config QA stage job

Stage Job 에 Azure App Service deploy 태스크를 추가합니다.

Add tasks | Refresh

azure app

-  **Azure App Service manage**
Start, stop, restart, slot swap, slot delete, install site extensions or enable continuous monitoring for an Azure App Service
-  **Deploy Azure Static Web App**
[PREVIEW] Build and deploy an Azure Static Web App
-  **Azure App Service deploy**
Deploy to Azure App Service a web, mobile, or API app using Docker, Java, .NET, .NET Core, Node.js, PHP, Python, or Ruby
-  **Azure Web App for Containers**
Deploy containers to Azure App Service
-  **Azure Web App**
Deploy an Azure Web App for Linux or Windows

태스크 속성은 아래와 같이 지정 또는 설정합니다. **Azure Subscription** 을 지정하기 위해서는 **6.2 Service Connection** 항목을 참조하시기 바랍니다.

Azure App Service deploy ⓘ

[View YAML](#) [Remove](#)

Task version 4.*

Display name *

Azure App Service Deploy: dotnetapp

Connection type * ⓘ

Azure Resource Manager

Azure subscription * ⓘ | [Manage](#)

MSET subscription

App Service type * ⓘ

Web App on Windows

App Service name * ⓘ

ikwebapp

☒ Deploy to Slot or App Service Environment ⓘ

Resource group *

test-vmss

Slot * ⓘ

qa

Virtual application ⓘ

Package or folder * ⓘ

\$(System.DefaultWorkingDirectory)/**/*.zip


팁: Build pipeline 의 BuildId 등의 내부적으로 사용되는 환경변수는 system.debug 변수를 true 로 변경하여 확인이 가능

5.4 Config Prod stage job

Stage Job 에 Azure Web App on Container 태스크를 추가합니다.

Add tasks | Refresh

swap

 **Azure App Service manage**
Start, stop, restart, slot swap, slot delete, install site extensions or enable continuous monitoring for an Azure App Service

태스크 속성은 아래와 같이 지정 또는 설정합니다.

Azure App Service manage ⓘ

[View YAML](#) [Remove](#)

Task version 0.* ▼

Display name *

Swap Slots: dotnetapp

Azure subscription * ⓘ | [Manage](#)

MSFT subscription ▼ ↻

Action ⓘ

Swap Slots ▼

App Service name * ⓘ

lkwebapp ▼ ↻

Resource group * ⓘ

test-vmss ▼ ↻

Source Slot * ⓘ

qa ▼ ↻

☒ Swap with Production ⓘ

☐ Preserve Vnet ⓘ

5.5 Trigger release

릴리즈를 트리거하여 배포를 진행합니다.

The screenshot shows the Azure DevOps Release Pipeline interface for a pipeline named 'Release-pipeline-bg' and a specific release named 'Release-3'. The interface is divided into two main sections: 'Release' and 'Stages'.

Release Section:

- Manually triggered** by Il Joong Kim on 2/17/2022, 11:52 AM.
- Artifacts:** A list showing the artifact '_dotnetapp_aspnet_ci' with version '20220126.1' and a 'master' branch.

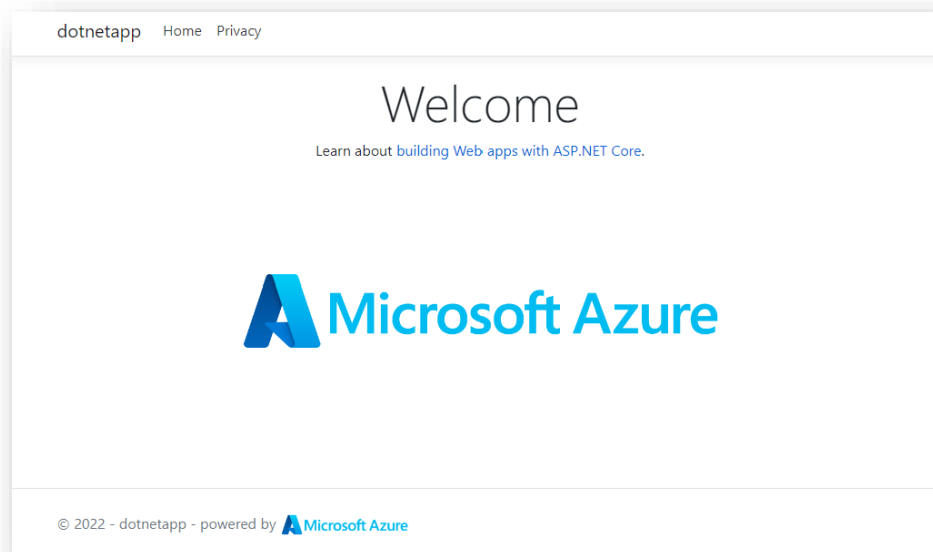
Stages Section:

- Deploy to QA:** Status 'Succeeded' (green checkmark), completed on 2/17/2022, 11:53 AM.
- Swap QA/Prod:** Status 'Succeeded' (green checkmark), completed on 2/17/2022, 11:56 AM. It also shows '1 warning'.

Deploy 시간은 Release 대쉬보드에서 완료된 시간과 차이가 발생합니다.

5.6 Test deployment

Web App의 URL로 접속하여 확인 정상적으로 배포되었는지 확인합니다.



6 Appendix

6.1 Git clone/Credential

Windows 환경이 아닌 Linux 에서 Azure devops 의 Source repo 를 로컬 PC 에 clone 할 때 Credential 정보를 확인

Git config 를 통해 credential 을 local 에 저장

```
git config credential.helper store
```

6.2 Service Connection

Azure 연동을 위한 Azure Subscription *Service Connection* 을 생성합니다.

Project settings->*Service connections*->**[Create/New service connection]**, *Azure Resource Manager* 선택하고 *[Next]*를 클릭합니다.

New service connection

Choose a service or connection type

Search connection types

- ☐ Azure Classic
- ☐ Azure Repos/Team Foundation Server
- ☒ Azure Resource Manager

인증 방식은 추천 방식을 선택하고 **[Next]**를 클릭합니다.

New Azure service connection

Azure Resource Manager

Authentication method

- ☒ Service principal (automatic) Recommended
- ☐ Service principal (manual)
- ☐ Managed identity
- ☐ Publish Profile

마지막으로, 연동할 구독을 선택하고 *service connection* 명을 지정하고 저장합니다.

New Azure service connection

Azure Resource Manager using service principal (automatic)

Scope level

- ☒ Subscription
- ☐ Management Group
- ☐ Machine Learning Workspace

Subscription

Azure [redacted] Subscription ([redacted]).

Resource group

[redacted]

Details

Service connection name

MyAzureSubscription

Description (optional)

[redacted]

Security

- ☒ Grant access permission to all pipelines