



DevOps Hackathon

Azure DevOps for Container

Version 1.0 Final

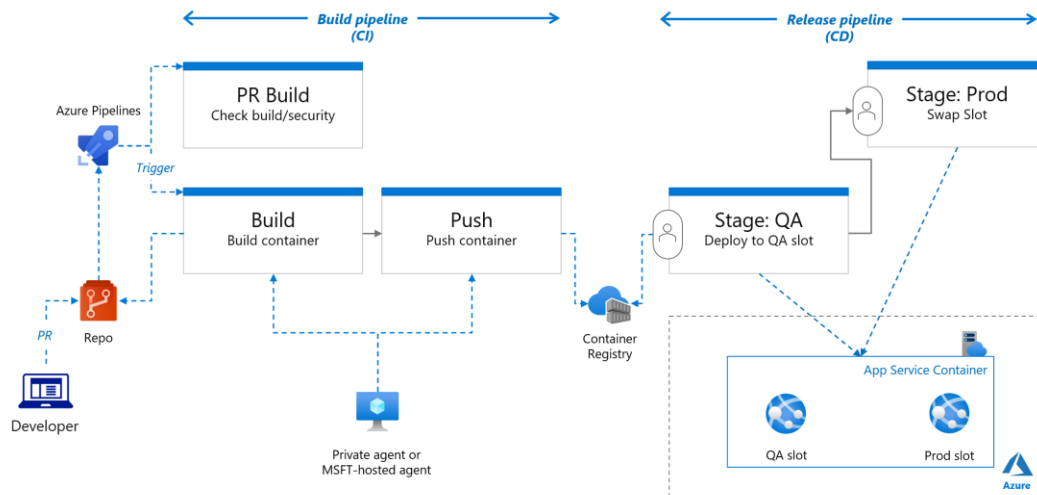
Table of Contents

1	Introduction	3
2	Challenge Overview	4
3	DevOps setup	5
3.1	Create new project	5
3.2	Setup Repo	6
3.3	Config self-hosted agent (Optional)	6
3.4	Prepare ACR	8
3.5	Service Connection	8
4	Challenge 1: Build pipeline (Classic)	10
4.1	Config build pipeline	10
4.2	Queue build	13
5	Challenge 2: Build pipeline (YAML)	14
5.1	Edit build pipeline	14
5.2	Build pipeline using ACR Build (Optional)	15
5.3	Setup pipeline trigger	16
6	Challenge 3: Release pipeline	18
6.1	Setup target VM(App Service/Web App)	18
6.2	Config release pipeline	19
6.3	Config QA stage job	20
6.4	Config Prod stage job	22
6.5	Trigger release	22
7	Challenge 4: Approval & branch policy	24
7.1	Approval	24
7.2	Branch policy (optional)	25

1 Introduction

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

CI/CD pipeline (container)



챌린지 내용:

Challenge	Description	Note
	DevOps Setup	
1	Build pipeline (Classic)	
2	Build pipeline (YAML)	
3	Release pipeline (UI)	
4	Extra: Approval and more	

2 Challenge Overview

Azure DevOps Workshop 의 챌린지는 다음과 같습니다.

Challenge 1: Build pipeline (classic)

UI 기반의 Classic pipeline 기능을 이용하여 기본적인 Build pipeline 을 구현합니다.

Challenge 2: Build pipeline (YAML)

코드로 관리하기 용이한 Yaml 기반의 pipeline 기능을 이용하여 Build pipeline 을 구현합니다.

Reference:

- <https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/key-pipelines-concepts?view=azure-devops>
- <https://learn.microsoft.com/en-us/azure/devops/pipelines/yaml-schema/?view=azure-pipelines>
- <https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/yaml-pipeline-editor?view=azure-devops>

Challenge 3: Release pipeline (UI)

Yaml 기반의 pipeline 으로 Release pipeline 구성도 가능하나, 좀더 손쉽게 구성할 수 있는 Azure DevOps 의 Release 기능을 이용하여 Release pipeline 을 구현합니다.

Reference:

- <https://learn.microsoft.com/en-us/azure/devops/pipelines/release/define-multistage-release-process?view=azure-devops>

Challenge 4: Approval, PR policy and Notification

Release pipeline 에 Approval 을 추가하고, Branch 보호를 위한 PR policy 추가합니다.

Azure DevOps with Web App for Container Reference:

<https://learn.microsoft.com/en-us/azure/devops/pipelines/apps/cd/deploy-docker-webapp?view=azure-devops&tabs=dotnet-core%2Cyaml>

3 DevOps setup

사전준비 사항: Azure 구독 (구독 Owner 권한 권장)과 Azure DevOps 계정이 필요. 본 실습은 Self-hosted (private) agent 를 권장하나 Microsoft hosted agent 에서도 진행 가능.

Azure DevOps lab 을 진행하기에 앞서 필요한 리소스들을 미리 준비합니다.

- 신규 Azure DevOps 계정을 준비 또는 생성
- Self-hosted agent 생성 (옵션)
 - Linux VM (Ds4_v3 권장) 생성
 - Agent pool 생성 (예: *agentpool*) 및 PAT 토큰 생성 ([agent 권한](#))
 - 개발도구 설치
 - Agent VM 에 개발 관련 도구 설치: .NET Core SDK, Docker, Azure CLI 등
 - Agent 설치 (상세 내용은 2.2 참조)
- 배포 대상 Cluster 및 추가 리소스 생성
 - App Service (Default configuration)
 - ACR (Standard SKU)

3.1 Create new project

Azure DevOps 계정에서 **[+ New project]**를 클릭하여 신규 프로젝트를 생성합니다.

Create new project ×

Project name *
devopscontainer

Description
Azure DevOps Workshop for Container

Visibility

<input type="radio"/> Public Anyone on the internet can view the project. Certain features like TFVC are not supported.	<input checked="" type="radio"/> Private Only people you give access to will be able to view this project.
--	---

▼ Advanced

프로젝트 생성 후 *Project Settings->Teams* 이동하여 HoL 을 함께 진행할 프로젝트 멤버를 추가합니다. (멤버 추가전에 *Organization settings->General/Users* 에서 멤버 초대/추가)

3.2 Setup Repo

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



Import a Git repository

Repository type

Git

Clone URL *

<https://github.com/iljoong/devops-container>

☐ Requires Authentication

추가 옵션: Import 방식대신 Local PC 로 소스를 다운 받아 리포에 Push 하는 것도 권장

소스가 준비되면 Azure DevOps 에 생성된 Source 폴더 구조와 내용을 리뷰 합니다.

3.3 Config self-hosted agent (Optional)

먼저 VM 을 생성하고 개발 관련 도구를 설치합니다.

- Docker 및 필요 Tool (.NET SDK 등) 설치 (.NET SDK 는 pipeline 에서 설치 가능)
 - **.NET SDK:** <https://dotnet.microsoft.com/download>
- Docker 커맨드 권한 설정 (`sudo usermod -aG docker $USER`)

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 type:

Self-hosted

A pool of agents that you set up and manage on your own to run jobs. [Learn more](#).

Name:

agentpool

Description (optional):

linux pool

[Markdown supported.](#)

Pipeline permissions:

☒ Auto-provision this agent pool in all projects

앞서 생성한 pool 을 선택하고 **[New agent]** 클릭합니다. 화면에 표시된 설치 스크립트를 복사 후, Agent VM 에서 (서비스로 실행 시 "svc.sh" 사용) 실행합니다.

Get the agent

Windows macOS **Linux**

x64
ARM
ARM64
RHEL6

System prerequisites

Configure your account
Configure your account by following the steps outlined [here](#).

Download the agent
[Download](#)

Create the agent

```
~/ $ mkdir myagent && cd myagent  
~/myagent$ tar zxvf ~/Downloads/vsts-agent-linux-x64-3.225.0.tar.gz
```

Configure the agent [Detailed instructions](#)

```
~/myagent$ ./config.sh
```

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

```
~/myagent$ ./run.sh
```

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

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

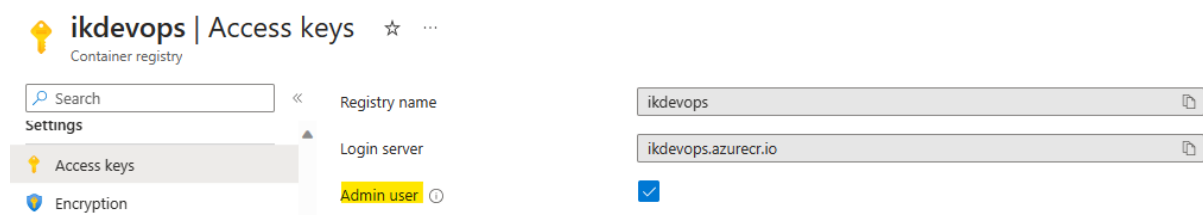
Ubuntu 20.04 Hosted agent: <https://github.com/actions/runner-images/blob/main/images/linux/Ubuntu2004-Readme.md>

세부 설치 내용은 아래 문서링크를 참조하시기 바랍니다.

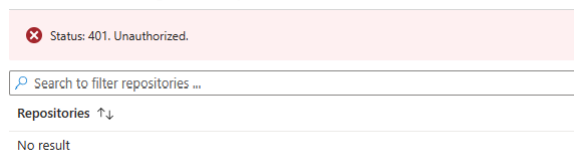
<https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/linux-agent?view=azure-devops>

3.4 Prepare ACR

컨테이너 이미지 빌드를 저장할 Azure Container Registry (ACR)를 먼저 생성합니다. ACR 생성 후 아래와 같이 Access Key 설정에서 Admin user 를 활성화합니다.



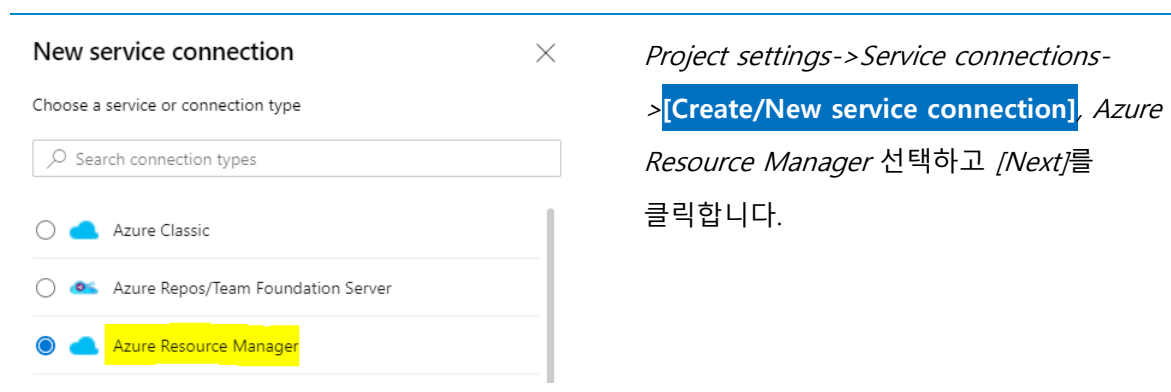
ACR 의 Data plane 접근 정책이 변경되어 기본 Role 로는 권한이 없어 Repositories 내의 정보를 확인할 수 없습니다.



Repositories 정보를 확인하려면 사용자에게 AcrPull Role 을 추가해야 합니다. (참조: <https://learn.microsoft.com/en-us/azure/container-registry/container-registry-roles>)

3.5 Service Connection

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



New Azure service connection

Azure Resource Manager

Authentication method

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

인증 방식은 추천 방식을 선택하고 [Next]를 클릭합니다.

New Azure service connection

Azure Resource Manager using service principal (automatic)

Scope level

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

Subscription

Microsoft Azure [redacted] ...

Resource group

[redacted]

Details

Service connection name

AzureSubscription

Description (optional)

Azure Subscription

Security

☒ Grant access permission to all pipelines

[Learn more](#)

[Troubleshoot](#)

Back

Save

마지막으로, 연동할 구독을 선택하고 *service connection* 명을 지정(예: AzureSubscription)하여 저장합니다.

인증을 위해 브라우저의 Pop-up 창을 활성화 필요 (새로 시작 필요)

Docker Registry(ACR) 연결을 위한 추가 Service Connection 을 진행하고 앞서 생성한 ACR 을 지정합니다.

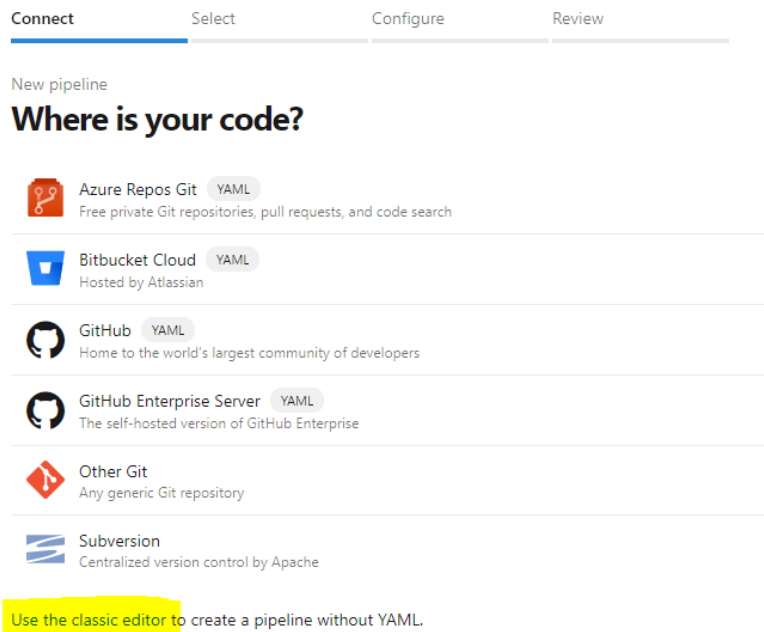
☐ Docker Host

☒ Docker Registry

4 Challenge 1: Build pipeline (Classic)

4.1 Config build pipeline

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





노트: Classic editor 가 활성화되어 있지 않은 경우에는 *Organization settings->Pipelines->settings* 로 이동하고 **Disable creating of class build pipelines** 과 **Disable creating of class release pipelines** 설정으로 Off 로 설정.


- ☐ Off **Disable creation of classic build pipelines**
No classic build pipelines can be created / imported. Existing ones will continue to work.
- ☐ Off **Disable creation of classic release pipelines**
No classic release pipelines, task groups, and deployment groups can be created / imported. Existing ones will continue to work.


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


Select a source


 Azure Repos Git

 GitHub

 GitHub Enterprise Server

 Subversion

 Bitbucket Cloud

 Other Git

Team project
aksdevops

Repository
aksdevops

Default branch for manual and scheduled builds
main


Continue

Template 선택에서 Docker 를 검색하여 선택하여 완료합니다.


Select a template

Or start with an [Empty job](#)

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](#)

Featured

 **Docker container**
Build a Docker image and push it to a container registry.

기본적으로 아래와 같이 *build an image, push an image* 태스크가 생성됩니다. 파이프라인 이름은 **"classicbuild-pipeline"**으로 변경하고 Agent pool 은 Microsoft host agent/Azure Pipeline 및 ubuntu-latest 를 선택하거나 앞서 생성한 self-hosted agent pool 을 선택합니다.

... > classicbuild-pipeline

Tasks Variables Triggers Options History | Save & queue Discard Summary Queue ...

Pipeline
Build pipeline

Get sources
devopshol main

Agent job 1
Run on agent

Build an image
Docker

Push an image
Docker

Name *

classicbuild-pipeline

Agent pool ⓘ | Pool information | Manage

Azure Pipelines

Agent Specification *

ubuntu-latest

Parameters ⓘ

This pipeline doesn't have any pipeline parameters. Create them to share the most important settings between tasks and change them in one place.

[Learn more](#)

Build an image 태스크에서 아래의 화면과 같이 Azure subscription, Container Registry, Build Arguments, Additional Image Tags 설정 값을 추가합니다. Image Name 은 기본 $\$(Build.Repository.Name)$ 변수 대신 **webapp**으로 변경하여 지정합니다.

Pipeline
Build pipeline

Get sources
skaksdevops main

Agent job 1
Run on agent

Build an image
R Docker

Push an image
R Docker

Display name *

Build an image

Container Registry Type *

Azure Container Registry

Azure subscription | Manage

AzureSubscription

Scoped to subscription 'Microsoft Azure Internal Consumption (Main)'

Azure Container Registry

ikaksdevops

Action *

Build an image

Docker File *

**/Dockerfile

Build Arguments

BUILDID=\$(Build.BuildId)

Use Default Build Context

Image Name *

webapp\$(Build.BuildId)

Qualify Image Name

Additional Image Tags

Include Source Tags

Include Latest Tag

노트: 파이프라인에서 사용되는 변수를 추가 또는 값을 변경할 수 있음.

Push an image 태스크에서 아래의 화면과 같이 앞서 설정한 Azure Subscription, Container Registry, Build Arguments 을 추가합니다. 앞선 태스크와 마찬가지로, Image Name 은 기본 $\$(Build.Repository.Name)$ 변수 대신 **webapp**으로 변경하여 지정합니다.

Pipeline
Build pipeline

Get sources
skaksdevops main

Agent job 1
Run on agent

Build an image
R Docker

Push an image
R Docker

Display name *

Push an image

Container Registry Type *

Azure Container Registry

Azure subscription | Manage

AzureSubscription

Scoped to subscription 'Microsoft Azure Internal Consumption (Main)'

Azure Container Registry

ikaksdevops

Action *

Push an image

Image Name *

webapp\$(Build.BuildId)

Qualify Image Name

Additional Image Tags

Include Source Tags

Include Latest Tag

4.2 Queue build

빌드 파이프라인을 저장 및 빌드를 실행(queue)하고 Agent 의 log 를 확인합니다. **Enable system diagnostics** 를 체크하면 파이프라인의 디버깅 출력을 확인할 수 있습니다.

← Jobs in run #20230927.2
classicbuild-pipeline

Jobs

✓ Agent job 1	55s
Initialize job	<1s
Checkout devopsweba...	2s
Build an image	31s
Push an image	19s
Post-job: Checkout de...	<1s
Finalize Job	<1s
Report build status	<1s

✓ Build an image

```
1 Starting: Build an image
2 =====
3 Task      : Docker
4 Description : Build, tag, push, or run Docker images, or run a Docker command
5 Version    : 0.225.1
6 Author     : Microsoft Corporation
7 Help       : https://docs.microsoft.com/azure/devops/pipelines/tasks/build/docker
8 =====
9 /usr/bin/docker pull mcr.microsoft.com/dotnet/aspnet:7.0
10 7.0: Pulling from dotnet/aspnet
11 7dbc1adf280e: Pulling fs layer
12 969d48310aaa: Pulling fs layer
13 2194c6af6861: Pulling fs layer
14 98003354b5af: Pulling fs layer
15 a5db88be08ca: Pulling fs layer
16 98003354b5af: Waiting
17 a5db88be08ca: Waiting
18 969d48310aaa: Verifying Checksum
19 969d48310aaa: Download complete
20 7dbc1adf280e: Verifying Checksum
21 7dbc1adf280e: Download complete
22 2194c6af6861: Verifying Checksum
23 2194c6af6861: Download complete
```

빌드가 정상적으로 완료되면 최종 빌드 이미지는 ACR 의 repository 로 퍼블리쉬 되며, ACR 에서 확인합니다.

노트: 퍼블리쉬된 빌드(컨테이너) 이미지를 로컬 PC 로 다운로드(pull)하여 정상적으로 실행되는지 검증하는 것을 권장.

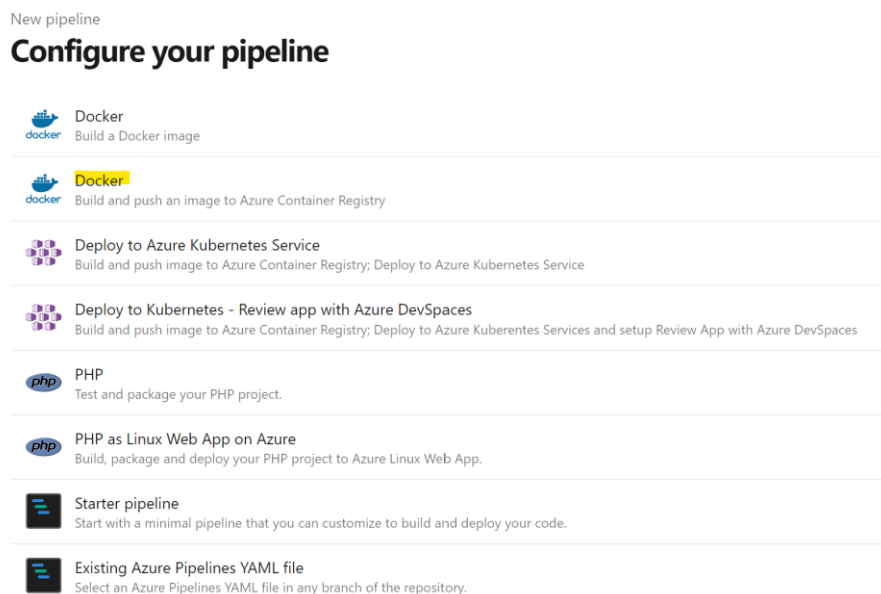
5 Challenge 2: Build pipeline (YAML)

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

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

5.1 Edit build pipeline

YAML 에서 Docker (Build and push an image to ACR) 템플릿을 선택합니다.



구독을 선택하고, 아래와 같이 관련 정보를 선택 또는 추가합니다.

Docker

Build and push an image to Azure Container Registry

Container registry

ikaksdevops

Image Name

webapp

Dockerfile

\$(Build.SourcesDirectory)/src/Dockerfile

최종 Review 단계에서 YAML 파일을 리뷰하고 저장합니다. YAML 파일을 **/pipelines/build-pipeline.yml** 로 저장합니다. Trigger 는 none 으로 변경하여 CI 트리거를 해제합니다.

New pipeline

Review your pipeline YAML

aksdevops / pipelines/build-pipeline.yml

```
1 # Docker
2 # Build and push an image to Azure Container Registry
3 # https://docs.microsoft.com/azure/devops/pipelines/languages/docker
4
5 trigger:
6 - none
7
8 resources:
9 - repo: self
10
```

기본 템플릿에서 사용되는 *buildAndPush* 태스크는 Docker arguments 를 사용할 수 없는 제약으로 소스 리포의 `./pipelines/_build-pipeline.yml` 파일을 참조하여 빌드 파이프라인을 수정합니다.

팁: 파이프라인 이름은 기본적으로 project 이름으로 생성되므로 빌드 명은 메뉴에서 원하는 이름으로 수정.

5.2 Build pipeline using ACR Build (Optional)

신규 파이프라인을 생성하고, **Starter pipeline** 템플릿을 선택합니다. 기본 내용은 삭제하고, 아래와 같은 YAML 형식으로 작성합니다. 화면 우측 **assistant** 의 기능을 이용하여 필요한 Task 를 검색하여 템플릿으로 추가할 수 있습니다.

노트: ACR 빌드 방법은 `./pipelines/_acrbuild-pipeline.yml` 파일을 참조.

aksdevops / pipelines/acrbuild-pipeline.yml * ❷

```
1 # Starter pipeline
2 # Start with a minimal pipeline that you can customize to build and deploy your code.
3 # Add steps that build, run tests, deploy, and more:
4 # https://aka.ms/yaml
5
6 trigger:
7   - none
8
9 pool:
10  - vmImage: ubuntu-latest
11
12 resources:
13   - repo: self
14
15 variables:
16   - # Container registry service connection established during pipeline creation
17     - dockerRegistryServiceConnection: '0c2b1108-a649-418a-bf68-5ca7f82c4815'
18     - imageRepository: 'webapp'
19     - containerRegistry: 'ikaksdevops.azurecr.io'
20     - dockerfilePath: '$(Build.SourcesDirectory)/src/Dockerfile'
21     - tag: '$(Build.BuildId)'
22
23 steps:
24   - task: AzureCLI@2
25     inputs:
26       azureSubscription: 'AzureSubscription'
27       scriptType: 'bash'
28       scriptLocation: 'inlineScript'
29       inlineScript: |
30         az acr build -r $(containerRegistry) -f $(dockerfilePath) \
31           --build-arg BUILDID=$(tag) \
32           -t $(containerRegistry)/$(imageRepository):$(tag) \
33           -t $(containerRegistry)/$(imageRepository):latest $(Build.SourcesDirectory)/src
```

Tasks

azure cli

 Azure CLI
Run Azure CLI commands against an Azure subscri...

Pipeline 파일명은 `/pipelines/acrbuild-pipeline.yml` 로 지정합니다. 파이프라인의 *variables* 을 지정합니다. 또한, ACR Build 를 위한 *AzureCLI* task 를 아래와 같이 추가합니다. 참고로, 앞서 생성한 빌드 파이프라인과 달리 Docker build 와 push 를 agent 에서 실행하는 것이 아닌 ACR 에서 직접 실행하는 빌드입니다.

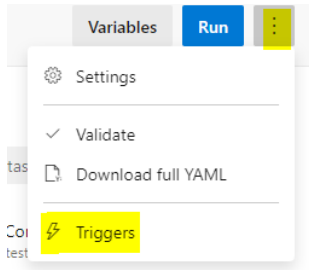
파이프라인 작성이 완료되면 앞서 실행했던 파이프라인과 동일하게 빌드를 실행하여 정상적으로 실행되는지 확인합니다.

5.3 Setup pipeline trigger

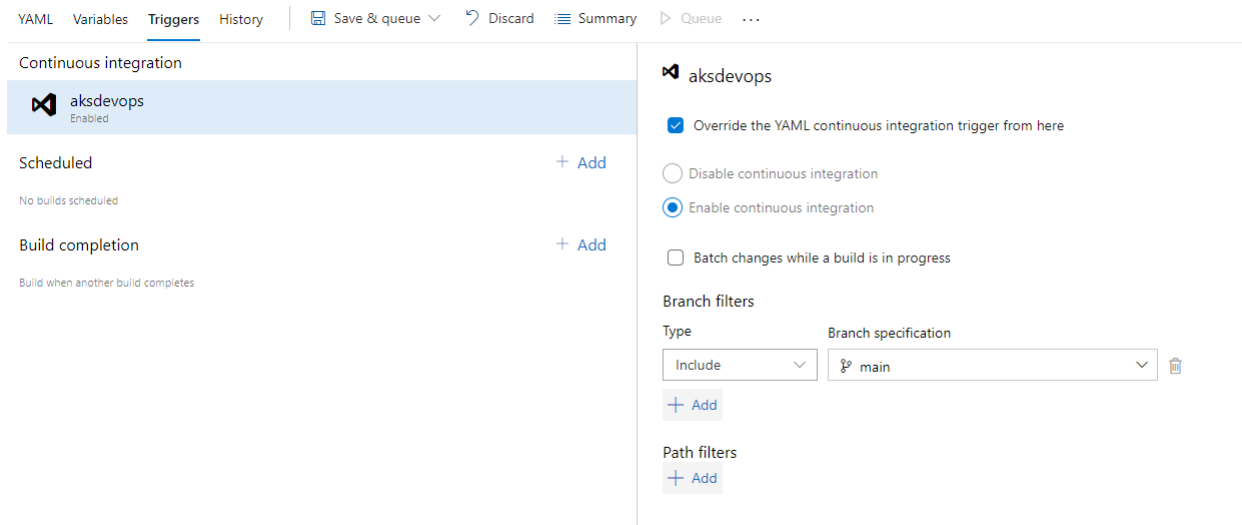
소스가 commit 되었을 때 빌드가 실행되는 Continuous Integration (CI)를 구성하고자 할 때 YAML 내에서 아래와 같이 trigger 를 편집합니다.

```
trigger:
  branches:
    include:
      - main
  paths:
    exclude:
      - pipelines/*
      - manifests/*
```

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



Classic 빌드 파이프라인과 동일한 UI 로 Triggers 탭에서 상세 설정이 가능합니다.



6 Challenge 3: Release pipeline

6.1 Setup target VM(App Service/Web App)

배포 타겟으로 아래의 설정으로 Web App 을 생성합니다.

Create Web App ...

Basics Docker 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 * ⓘ Visual Studio Enterprise Subscription ▼

Resource Group * ⓘ demo-devops-rg ▼

[Create new](#)

Instance Details

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

Name * ikweb ✓
.azurewebsites.net

Publish * ☐ Code ☒ Docker Container ☐ Static Web App

Operating System * ☒ Linux ☐ Windows

Region * Korea Central ▼

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

Pricing plans

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

Linux Plan (Korea Central) * ⓘ (New) ikweb-asp ▼

[Create new](#)

Pricing plan Premium V3 P1V3 (195 minimum ACU/vCPU, 8 GB memory, 2 vCPU) ▼

[Explore pricing plans](#)

기본 사이트(Production slot)를 생성 후, QA [Slot 을 미리 추가\(설정 복제/Clone Settings\)](#)합니다.

* QA slot 을 생성하기 위해서는 Standard 이상의 SKU 생성 필요

Docker 이미지는 앞서 빌드에서 배포한 ACR 의 이미지 및 태크를 지정합니다.

Create Web App ...

Basics **Docker** Networking Monitoring Tags Review + create

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options	Single Container
Image Source	Azure Container Registry
Azure container registry options	
Registry *	ikdevops
Image *	webapp
Tag *	200
Startup Command ⓘ	

6.2 Config release pipeline

신규 릴리즈 파이프라인(empty 템플릿)을 생성하고, 아래와 같이 파이프라인을 구성합니다.

All pipelines > Release-pipeline

Pipeline Tasks Variables Retention Options History

Save Create release View releases ...

Artifacts | + Add

Schedule not set

Stages | + Add

Deploy to QA 1 job, 1 task

Swap QA to Prod 1 job, 1 task

Artifact

Build - _build

Delete ...

Project *

devopswebapp

Source (build pipeline) *

classicbuild-pipeline

Default version *

Latest

Source alias *

_build

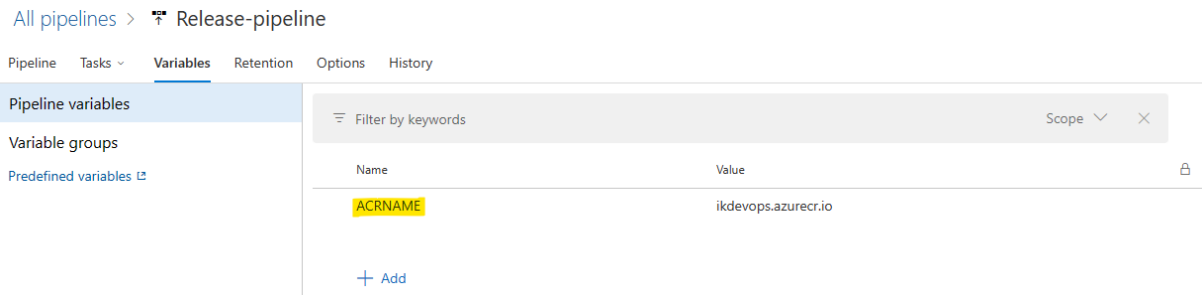
No version is available for classicbuild-pipeline or the latest version has no artifacts to publish. Please check the source pipeline.

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

Stages: QA 와 Prod 용 두개의 stage 생성

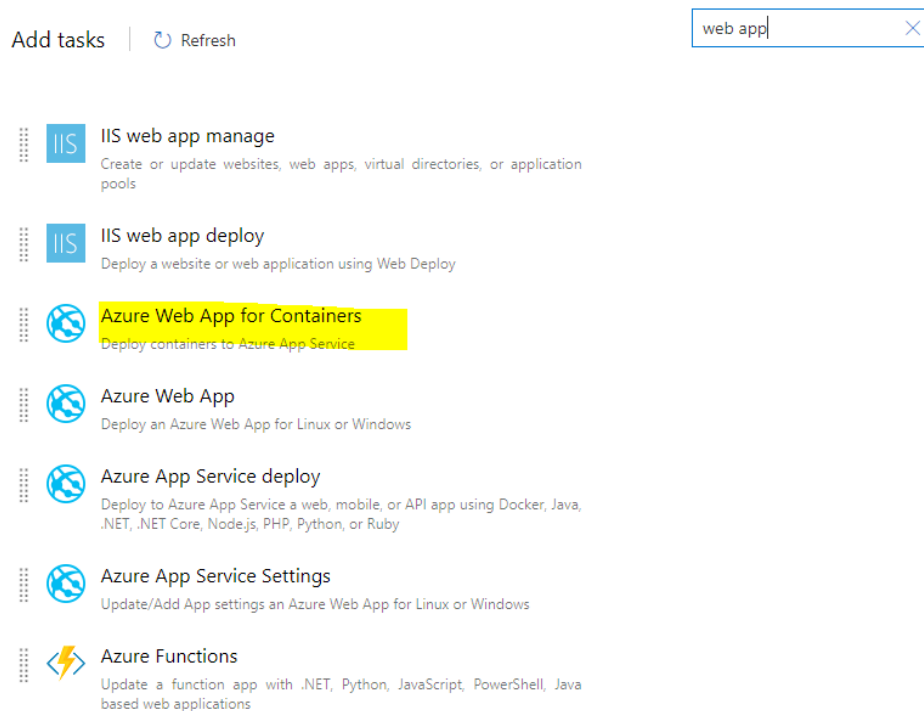
주의: Artifact 의 source alias 명에 hyphen('-') 이 들어가지 않게 수정. 예 _classicbuild-pipeline 는 _build 로 수정 (Linux 에서 환경변수가 정상적으로 설정 안됨)

릴리즈 파이프라인에서 사용될 변수를 *Variables* 편집창에 추가합니다.



6.3 Config QA stage job

Stage Job 에 Azure Web App on Container 태스크를 추가합니다.



태스크 속성은 아래와 같이 지정 또는 설정합니다.

Task version 1.* ▼

Display name *

Azure Web App on Container Deploy: ikweb

Azure subscription * ⓘ | [Manage](#) ↗

AzureSubscription ▼



ⓘ Scoped to subscription 'Visual Studio Enterprise Subscription'

App name * ⓘ

ikweb ▼

☒ Deploy to Slot or App Service Environment ⓘ

Resource group * ⓘ

demo-devops-rg ▼



Slot * ⓘ

production ▼



Image name ⓘ

`$(ACRNAME)/webapp:$(RELEASE.ARTIFACTS._BUILD.BUILDID)`

Configuration File ⓘ



Startup command ⓘ

이미지 이름은 아래를 참고하고, 모두 대문자로 작성합니다. (아래 `_BUILD` 변수는 앞서 설정한 빌드 artifact 의 source alias 임)

Image name: `$(ACRNAME)/webapp:$(RELEASE.ARTIFACTS._BUILD.BUILDID)`

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

6.4 Config Prod stage job

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

Stage

Job 에 Azure Web App on Container 태스크를 추가합니다.

태스크 속성은 아래와 같이 지정 또는 설정합니다.

Azure App Service manage ⓘ

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

Task version 0.* ▾

Display name *

Swap Slots: ikweb

Azure subscription * ⓘ | [Manage](#)

AzureSubscription ▾

ⓘ Scoped to subscription 'Visual Studio Enterprise Subscription'

Action ⓘ

Swap Slots ▾

App Service name * ⓘ

ikweb ▾

Resource group * ⓘ

demo-devops-rg ▾

Source Slot * ⓘ

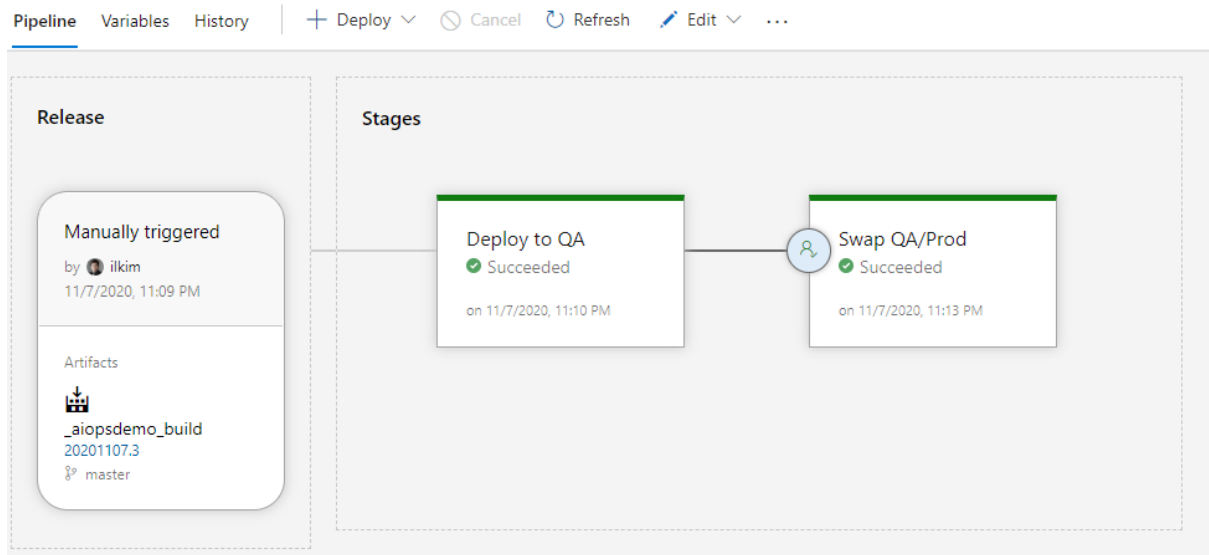
qa ▾

☒ Swap with Production ⓘ

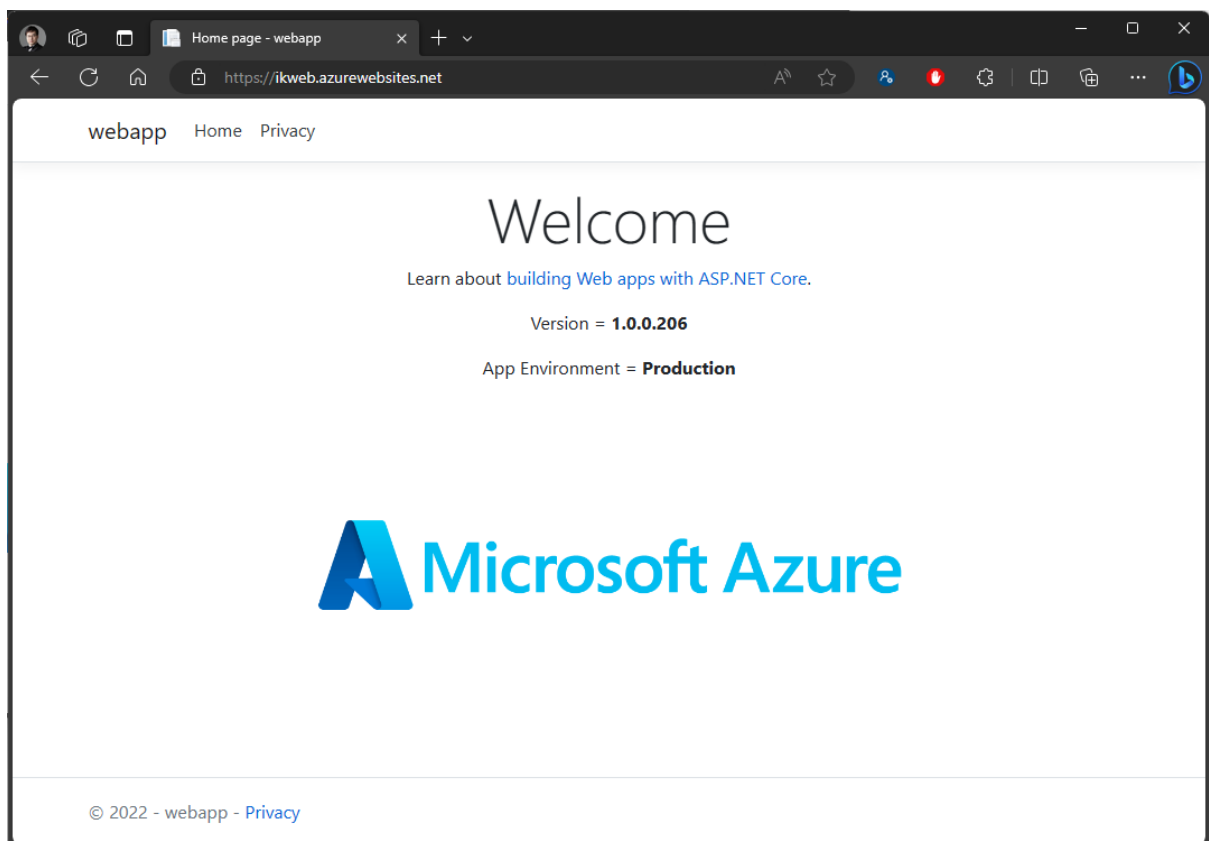
☐ Preserve Vnet ⓘ

6.5 Trigger release

릴리즈를 트리거하여 배포를 진행합니다.



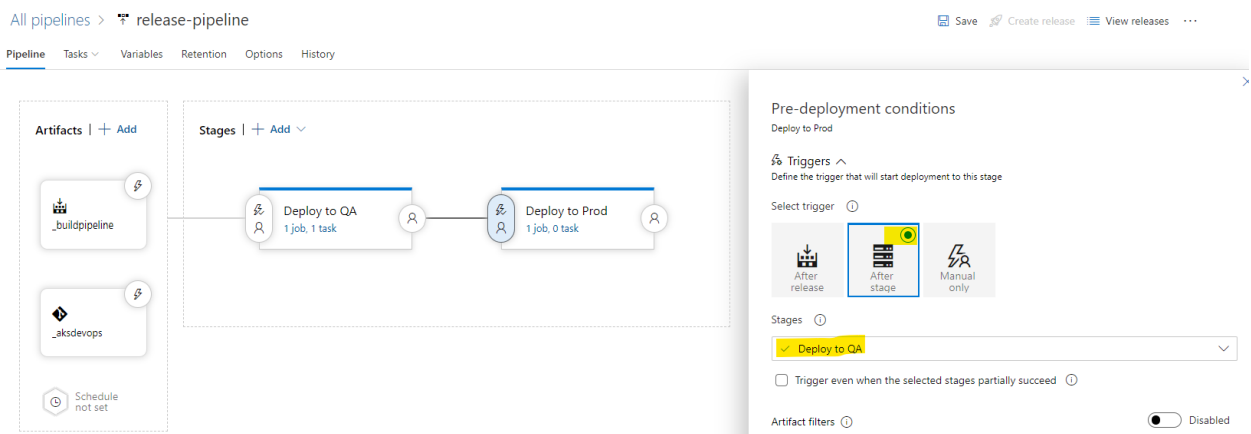
Deploy 시간은 Release 대쉬보드에서 완료된 시간과 차이가 발생합니다. 실제 Web App 의 적용시간은 docker image 다운로드를 포함하여 몇 분 소요될 수 있습니다.



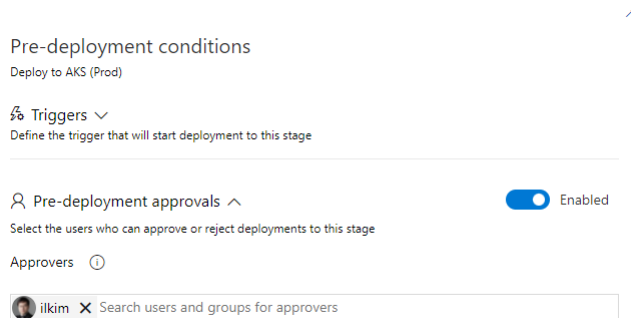
7 Challenge 4: Approval & branch policy

7.1 Approval

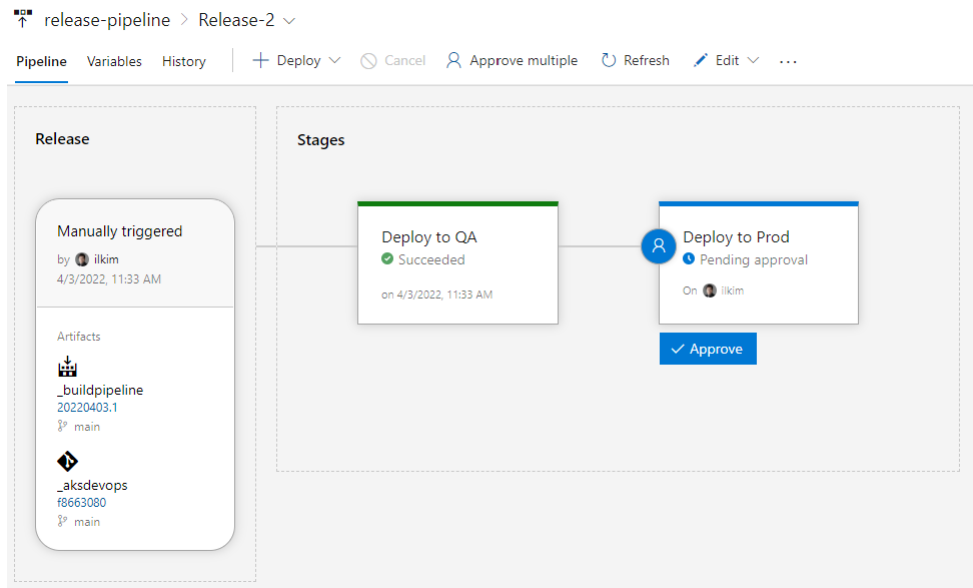
신규 스테이지(Deploy to Prod)를 추가하고 Trigger 를 아래와 같이 구성합니다. 또한, 앞서 QA Stage 의 태스크를 구성 것같이 동일하게 Prod stage 의 태스크를 구성을 합니다.



각 스테이지의 사전/사후 승인(approval)이 필요한 경우 아래와 같이 사람 아이콘(👤)을 클릭하여 승인자를 추가합니다.



빌드를 실행하여 릴리즈 파이프라인이 트리거하고 진행 현황을 확인합니다.



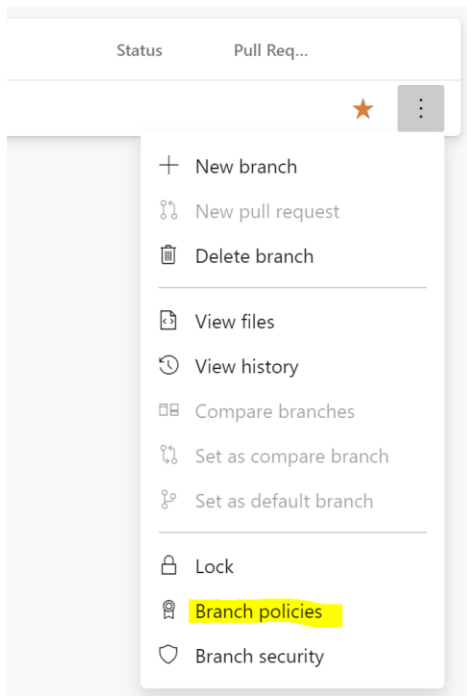
Web App 의 URL 로 접속하여 확인 정상적으로 배포되었는지 확인합니다.

7.2 Branch policy (optional)

소스코드의 빌드 품질을 높이는 방법의 하나로 Branch Policy 를 활용할 수 있습니다. 예를 들어, main branch 에 개발자의 코드를 직접 merge 하는 것이 아니라 PR(Pull Request)를 통해 정상적인 빌드와 리뷰어 확인후에만 merge 가 되도록 설정할 수 있습니다.

먼저, PR 빌드용 파이프라인을 생성하고 **pr-build-pipeline** 이름으로 지정합니다.

해당(master) branch 에서 Branch policies 를 선택하고, Build Validation **[+]**을 클릭합니다.



Build pipeline 항목에 앞서 생성한 pr-build-pipeline 를 선택합니다.

Add build policy ✕

Build pipeline *

pr-build-pipeline ▼

Path filter (optional)

ⓘ

Azure DevOps 에서 README.md 파일을 임의로 수정하고 Commit 을 수행합니다. PR 없이 main 브랜치에 commit 을 바로 할 경우 아래와 같은 에러가 출력됩니다.

Commit ✕

⊗ TF402455: Pushes to this branch are not permitted; you must use a pull request to update this branch.

Comment

Updated README.md

Branch name

main

신규 브랜치(예: test)를 생성하고 Program.cs 가 오류가 발생하도록 수정(';를 삭제)합니다.

```
.ConfigureWebHostDefaults(webBuilder =>
{
```

```
webBuilder.UseStartup<Startup>());  
});
```

Commit 후, PR 을 요청하면 아래와 같이 오류가 발생한 것을 확인할 수 있습니다.

Updated Program.cs

Active 11 ilkim test into main

Overview Files Updates Commits

1 required check failed

pr-pipeline Build failed

Re-queue

Job / DotNetCoreCLI

```
16 src/Program.cs(23,53): Error CS1002: ; expected  
26 Error: The process '/usr/bin/dotnet' failed with exit code 1  
29 Dotnet command failed with non-zero exit code on the following projects : /home/vsts/work/1/s/src/webapp.csproj
```

No merge conflicts

Last checked Just now