Microsoft

DevOps Hackathon

Azure DevOps for Windows/Deployment Group

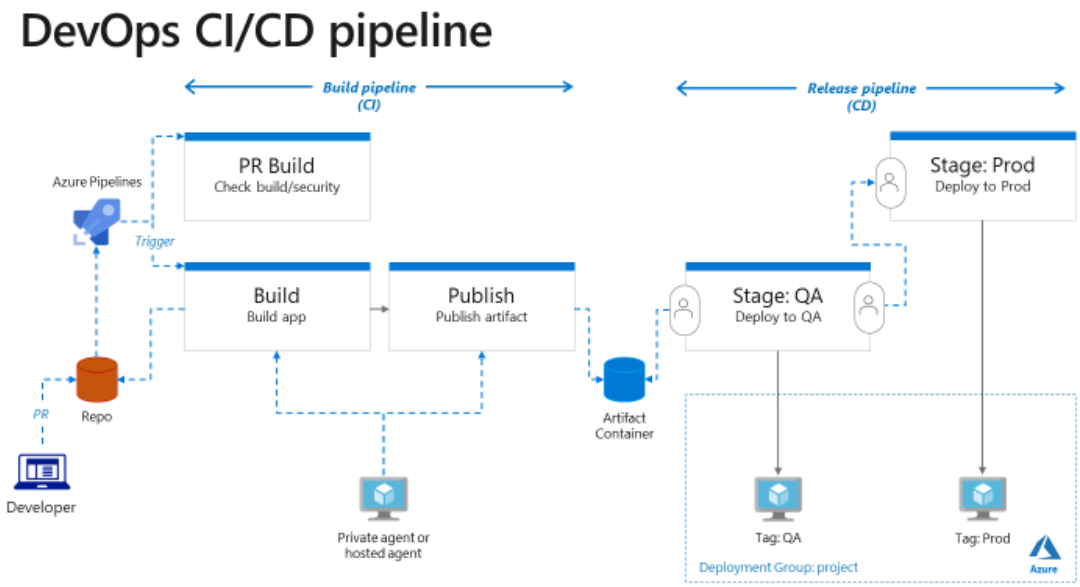
2023-10-09

Version 1.0 Final

Table of Contents

1 Introduction

본 DevOps Workshop 에서는 Azure DevOps 를 이용하여 .NET 기반의 DevOps CI/CD pipeline 만들어 봅니다.



챌린지 내용:

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

2 DevOps setup

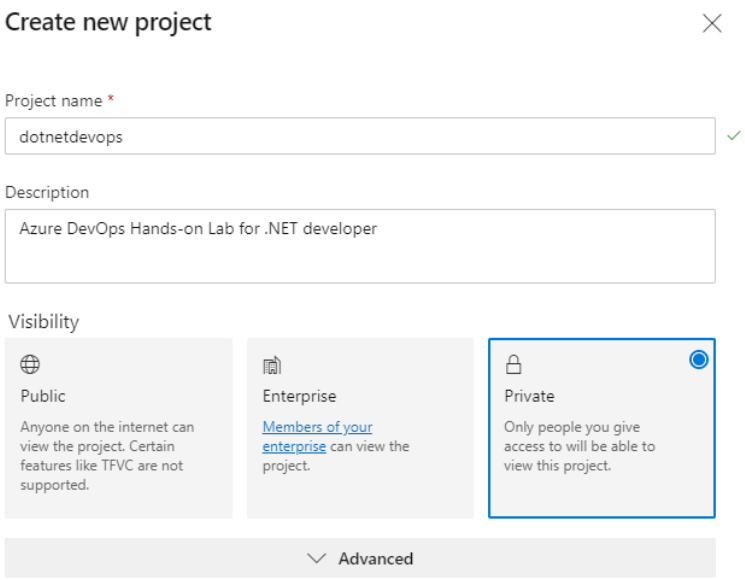
> **사전준비 사항:** Azure 구독 (구독 Owner 권한 권장)과 Azure DevOps 계정이 필요. 본 워크샵은 Azure DevOps 의 빌드/배포를 위해 최소 3 개의 VM 을 생성을 권장.

Azure DevOps lab 을 진행하기에 앞서 필요한 리소스들을 미리 준비합니다.

- 신규 DevOps 신규 Project 생성
  - Agent pool 생성 (예: *winpool*)
  - Deployment groups 생성
  - PAT 토큰 생성 (agent pool 권한)
- Private agent 생성
  - Windows OS VM (D2s_v3 또는 D2as_v4 사양 권장) 생성
  - 개발도구 설치
    - Agent VM 에 개발 관련 도구 설치: .NET Core SDK 등
  - Agent 설치 (상세 내용은 2.2 참조)
- 배포 대상 VM 생성
  - Windows Server 2019 이상 2 개의 VM 생성 (D2s_v3 또는 D2as_v4 사양 권장)
  - Deployment group agent 설치 (상세 내용은 5.3 참조)

2.1 Create new project

[New project]를 클릭하여 신규 프로젝트를 생성합니다.



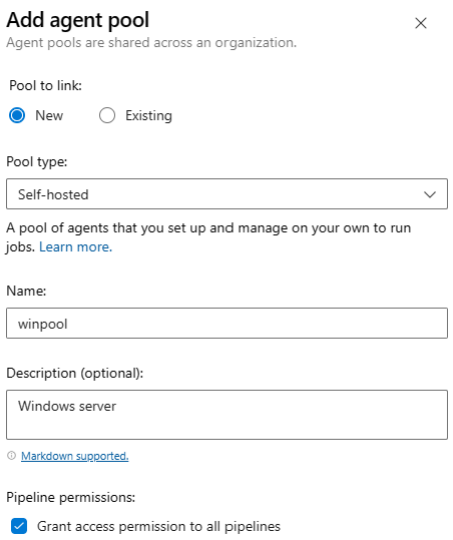
2.2 Config private agent

Privat agent 로 사용될 VM 을 생성하고, 개발 관련 도구(본 워크샵은 .NET 7 을 사용)를 설치합니다. (.NET SDK 는 pipeline 에서 설치 가능)

- **.NET SDK**: https://dotnet.microsoft.com/en-us/download/dotnet/7.0

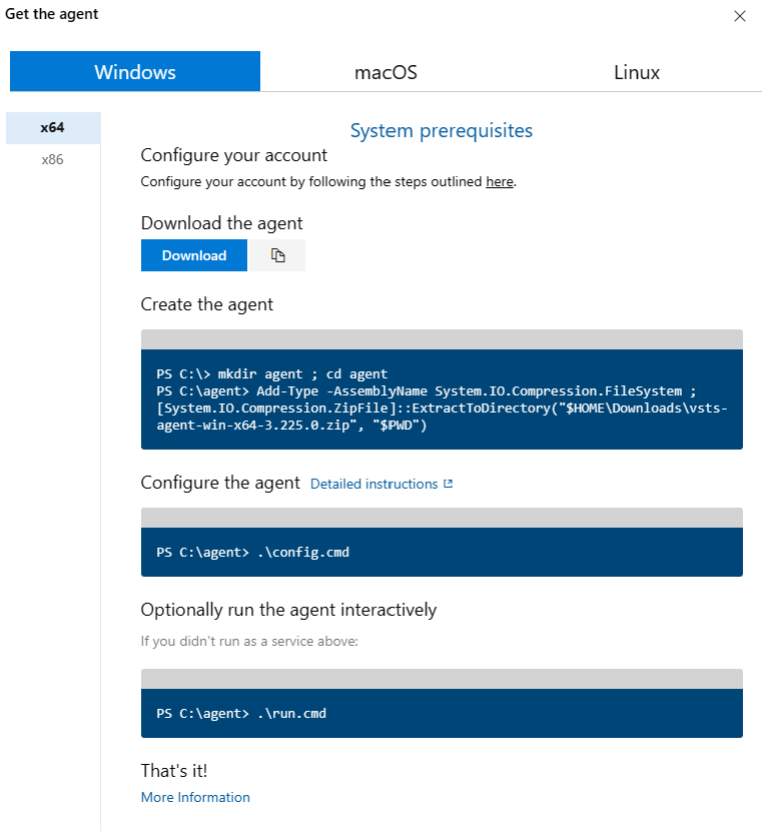
> - **팁:** 개발도구가 먼저 설치되어야 agent 가 정상적으로 인식.

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



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

> **노트:** 본 워크샵은 private agent 없이도 진행이 가능하지만, 향후 enterprise 환경에 좀더 적합한 DevOps 환경 구축을 위해 구성하는 것을 권장.
>
> **Microsoft Windows Server 2019 (agent 의 주요 설치 SW)**:
> https://github.com/actions/runner-images/blob/main/images/win/Windows2019-Readme.md



세부 설치 내용은 아래 문서링크를 참조하시기 바랍니다.

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

3 Challenge 개요

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

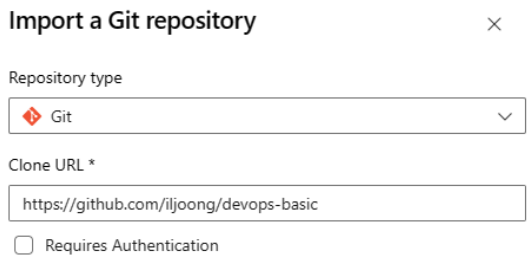
Release pipeline 에 Approval 을 추가하고, Branch 보호를 위한 PR policy 추가합니다.

Teams Notification 도 적용합니다.

4 Challenge 1: Build pipeline (Classic)

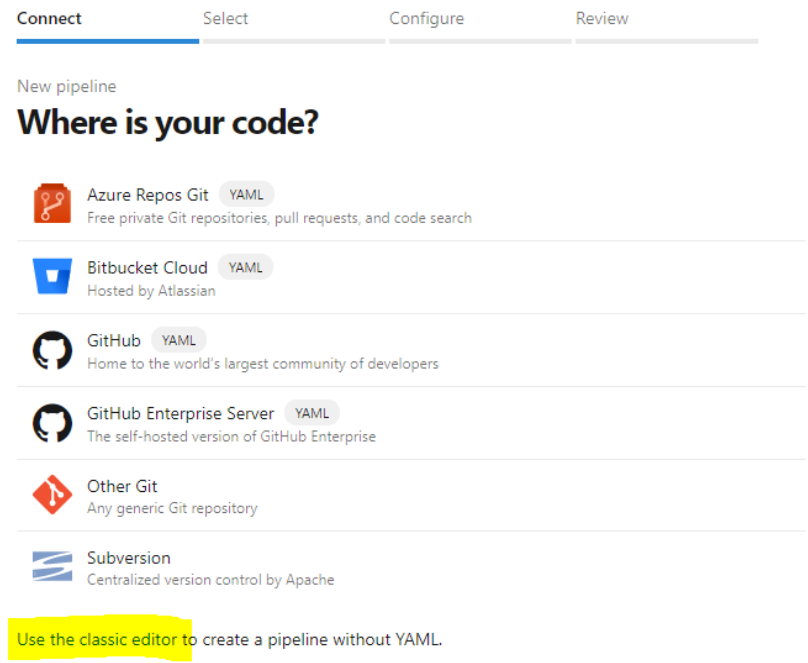
4.1 Setup Repo

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



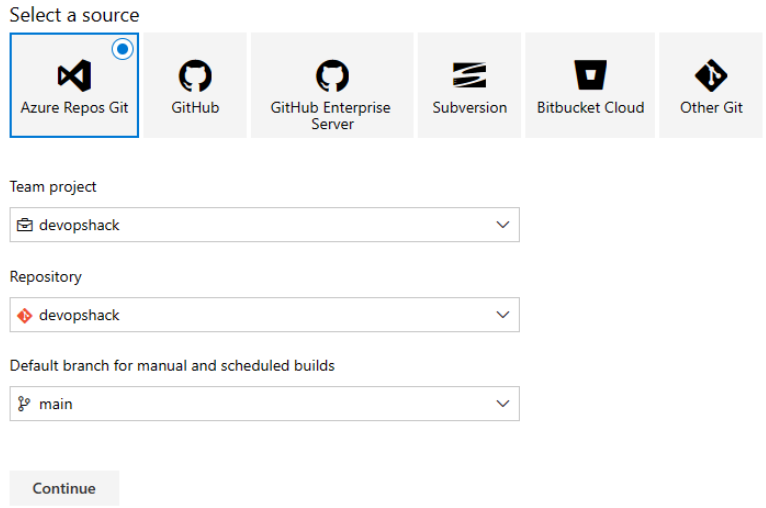
4.2 Config build pipeline

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

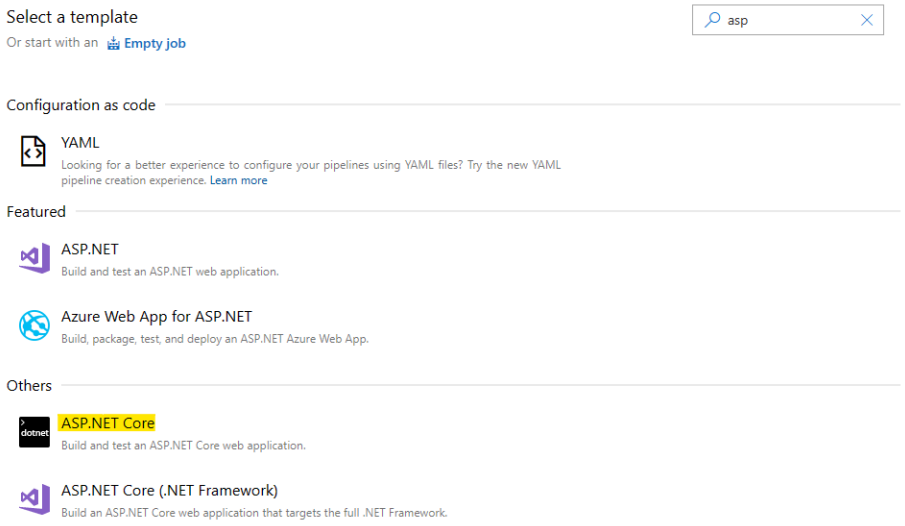


> **노트: Classic editor** 가 활성화되어 있지 않은 경우에는 *Organization settings->Pipelines->settings* 로 이동하고 **Disable creating of class build pipelines** 과 **Disable creating of class release pipelines** 설정으로 Off 로 설정.

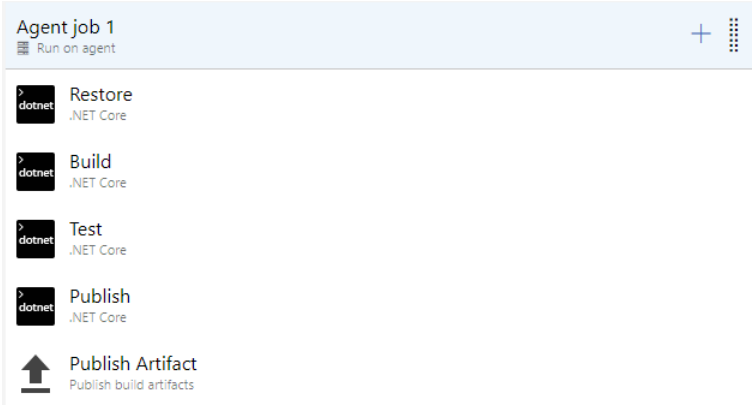
Source 선택에서 **Azure Repos Git** 을 선택하여 다음으로 이동합니다.



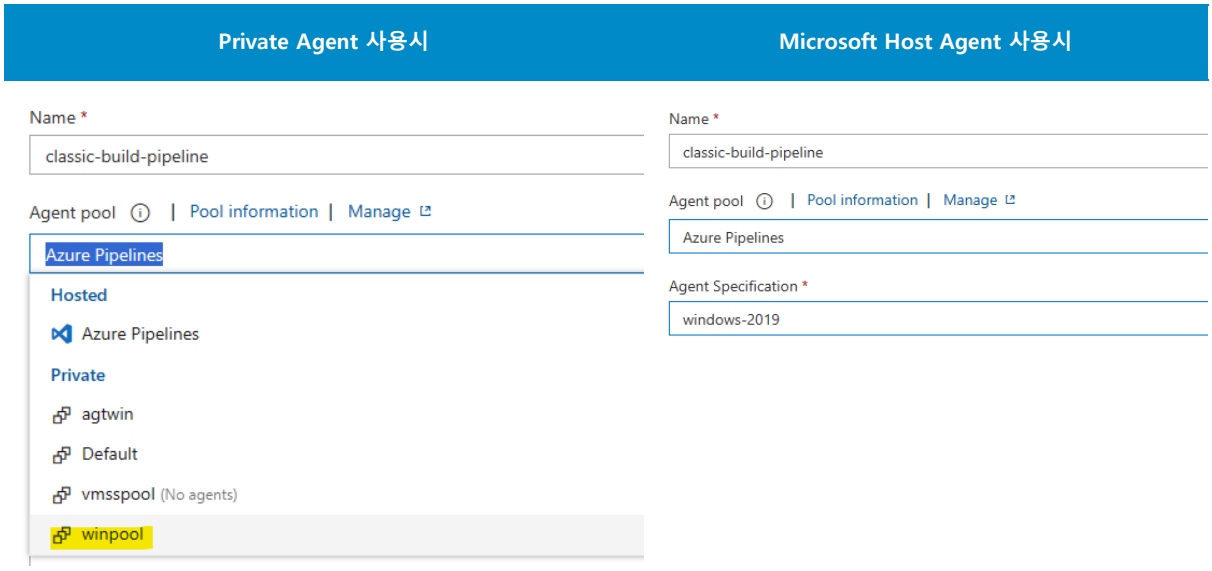
Template 선택에서 asp.net core 를 검색하여 선택하여 완료합니다.



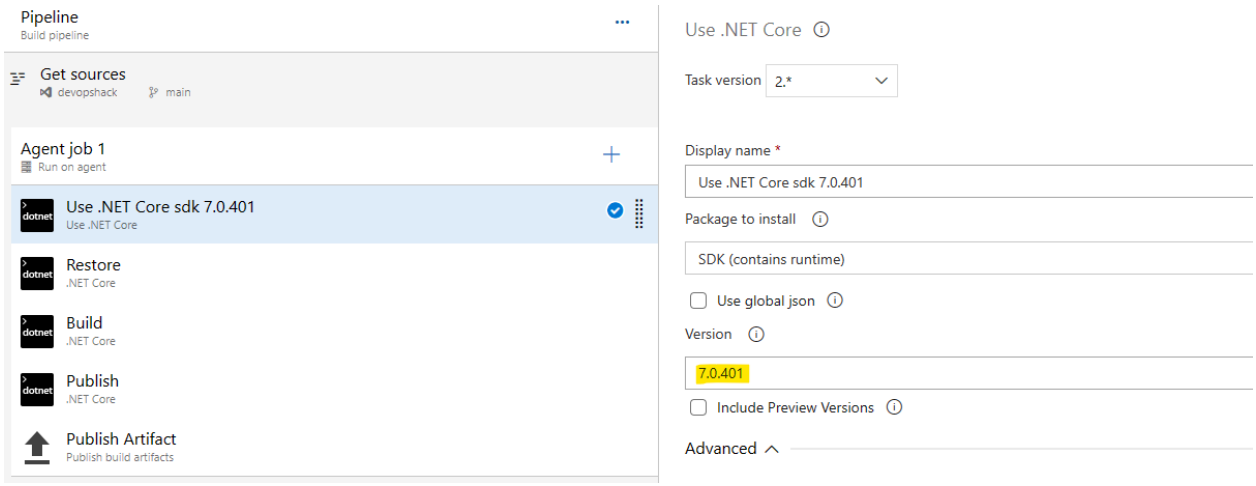
기본적으로 아래와 같이 ***restore, build, test, publish, publish artifact*** 태스크가 생성됩니다.



Agent pool 은 앞서 생성한 Private agent pool 을 선택하거나 *Microsoft host agent (windows-2019)*를 선택합니다.

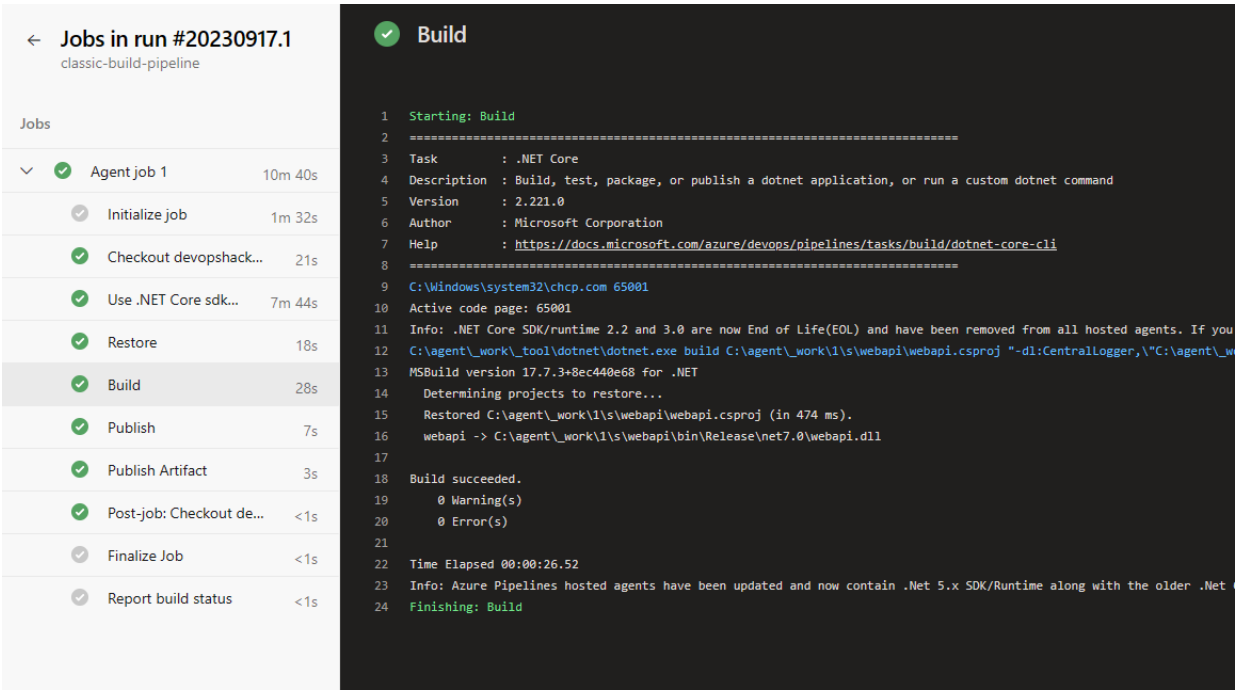


Default 로 생성된 Task 에서 *Test Task* 를 삭제하고, Agent Job 의 [+]를 클릭하여 ***Use .NET core Task*** 를 추가하고 SDK 버전을 지정합니다.

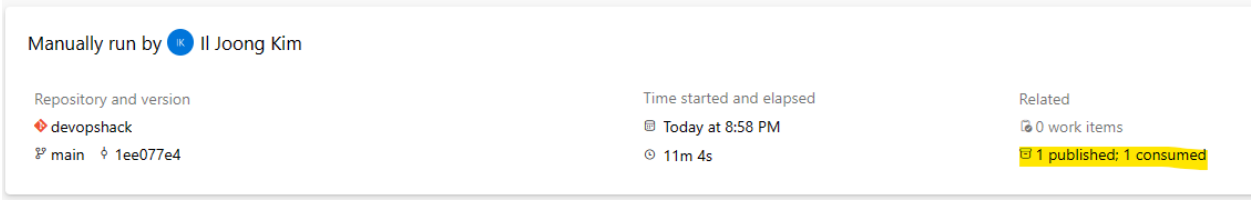


4.3 Queue build

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



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



> **노트:** Container 는 Azure DevOps 에서 제공되는 클라우드 내의 shared infrastructure 에 저장되며, private infrastructure 에 저장을 원할 경우 사용자 지정 file share 에 저장할 수 있음.

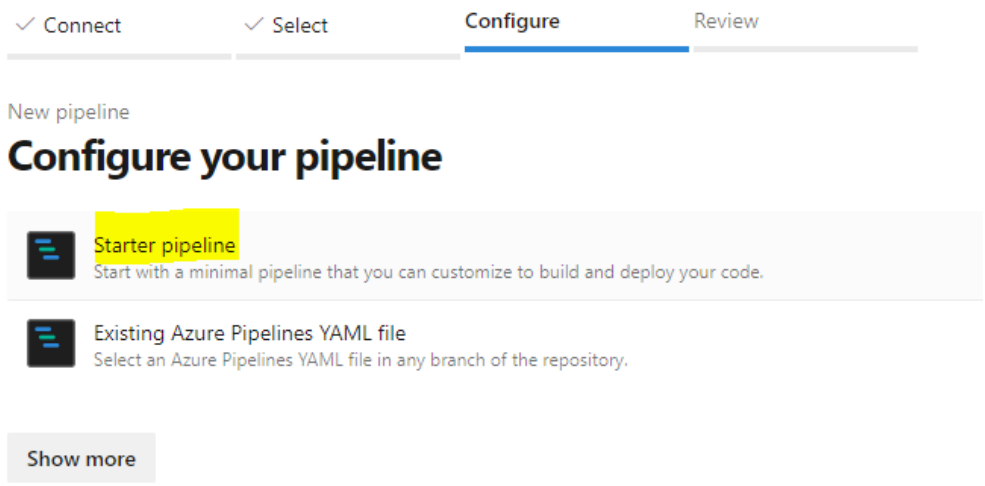
5 Challenge 2: Build pipeline (YAML)

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.

5.1 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. Pipeline 파일은 pipelines 폴더 하위에 저장합니다.

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

```
trigger: none

pool:
  name: '_your_pool_'
  #vmImage: windows-2019

variables:
```

```
- name: buildConfiguration
  value: 'Release'

steps:
- task: UseDotNet@2
  displayName: 'Use .NET sdk 7.0.401'
  inputs:
    version: 7.0.401
- 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.

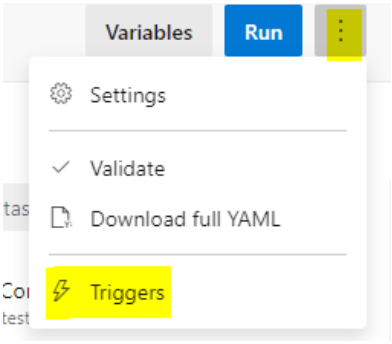
Pipeline 작성이 완료되면 앞서 실행했던 Classic pipeline 과 동일하게 빌드를 실행하여 정상적으로 실행되는지 확인합니다.

5.2 Setup pipeline trigger

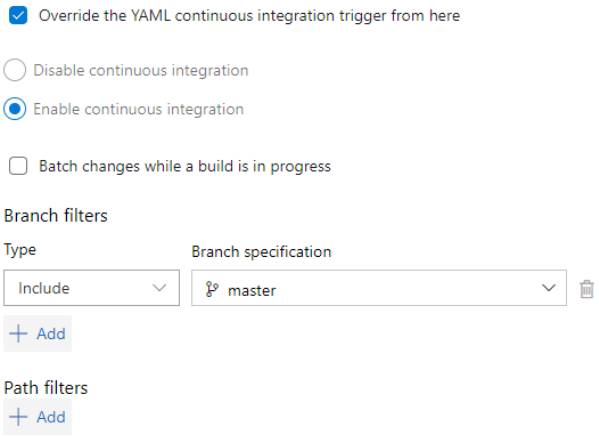
소스가 commit 되었을 때 빌드가 실행되는 Continuous Integration (CI)를 구성하고자 할 때 YAML 내에서 아래와 같이 trigger 를 편집합니다.

```
trigger:
- main
```

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.



6 Challenge 3: Release pipeline (UI)

6.1 Provision VM

릴리즈 파이프라인에 사용될 QA 와 Production 용 VM (Windows Server 2019 이상)을 Azure 에서 생성합니다. Azure 에서 Windows VM 생성방법에 대해서는 아래의 링크를 참조하시기 바랍니다.

https://learn.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-portal

6.2 Setup deployment VM (Windows)

.NET application 을 실행하기 위해서 서비스 등록을 위한 준비가 필요합니다.

- Windows OS IIS 설정
- .NET hosting runtime (7.0.11) 설치

파워셀 커맨드창을 열고 아래의 컨맨드를 실행합니다.

```
Add-WindowsFeature Web-Server,Web-Asp-Net45,NET-Framework-Features

Start-BitsTransfer
https://download.visualstudio.microsoft.com/download/pr/91644a20-1e21-43c9-8ae0-
90e402c1a368/469c198fab110c6c3d822e03509e9aec/dotnet-hosting-7.0.11-win.exe -
Destination $env:temp\dotnet-hosting-7.0.11-win.exe

Start-Process $env:temp\dotnet-hosting-7.0.11-win.exe -ArgumentList '/quiet' -Wait
```

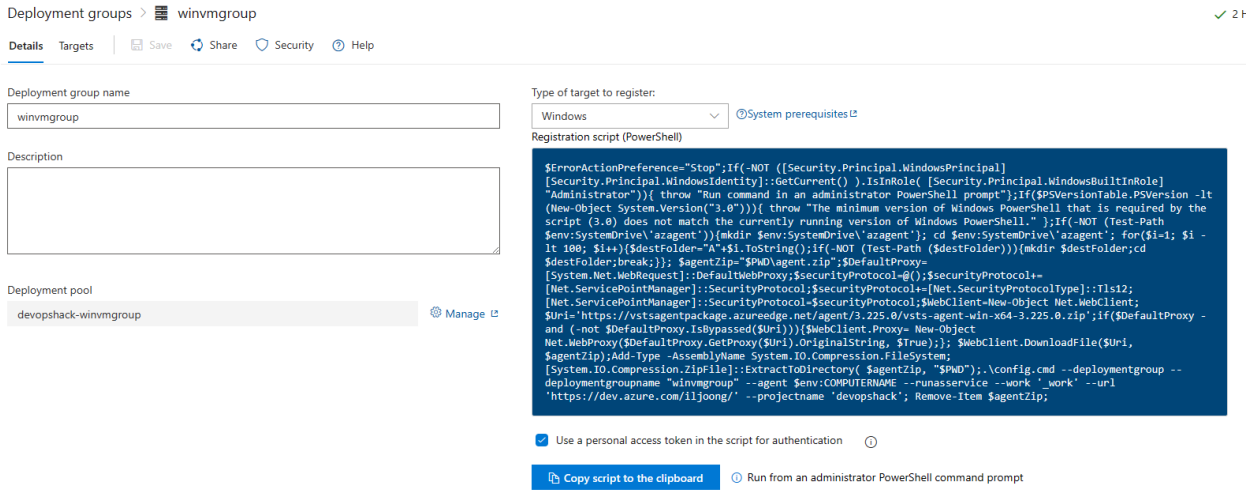
> **노트:** NSG 에서 Inbound rule 에 port 80(allow)을 추가합니다.

커맨드 실행이 완료되면 커맨드 창을 닫습니다. (신규 설치된 SW 인식을 위해)

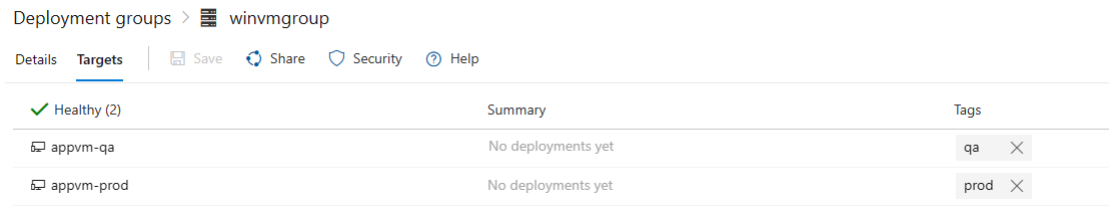
6.3 Config deployment groups VM

Pipelines->Deployment groups->**[Add a deployment group]**을 선택하여 신규 deployment group 을 생성합니다.

PAT (Personal Access Token) 체크박스를 활성화하고 화면의 설치 스크립트 복사하여 해당 VM 의 새로운 파워셀 커맨드창에서 실행합니다. 2 대의 VM 에 각각 실행하며, 스크립트 진행 중 VM #1 의 Tag 는 **QA**, VM #2 의 Tag 는 **Prod** 로 설정합니다.



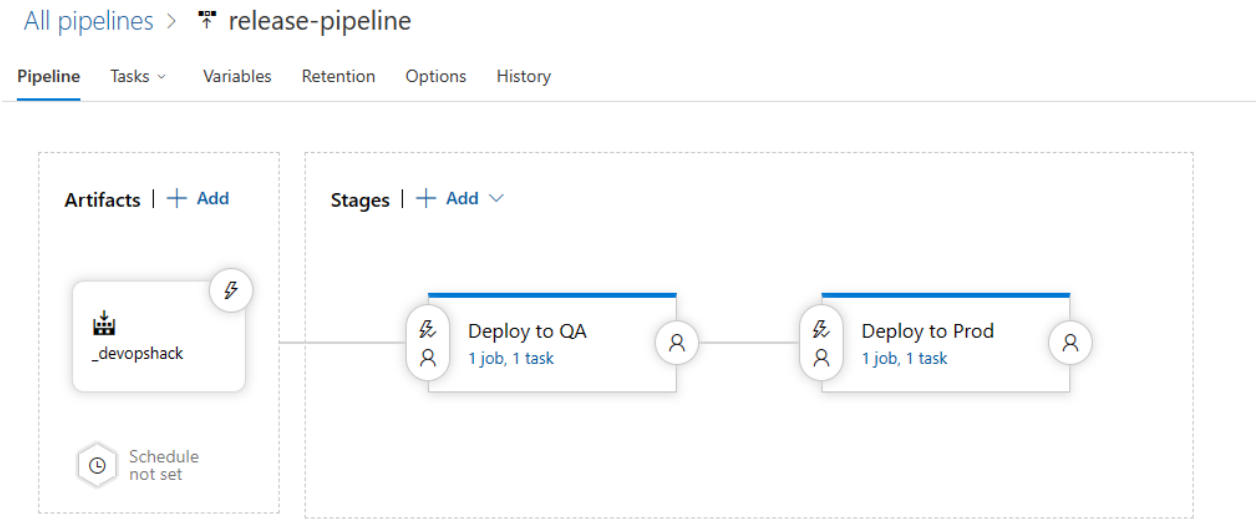
Agent 가 정상적으로 설치되면 Deployment group 에서 아래와 같이 VM 이 상태가 표시됩니다.



6.4 Config release pipeline

> **노트**: Deployment group 의 agent 는 private agent 와 유사하게 Azure DevOps 를 통해 빌드된 artifacts 를 Deployment Group 으로 연결된 VM 에 복사/다운로드 수행.

Pipelines->Deployment groups->[New pipeline]을 선택하여 아래와 같이 신규 릴리즈 파이프라인을 구성을 구성합니다.

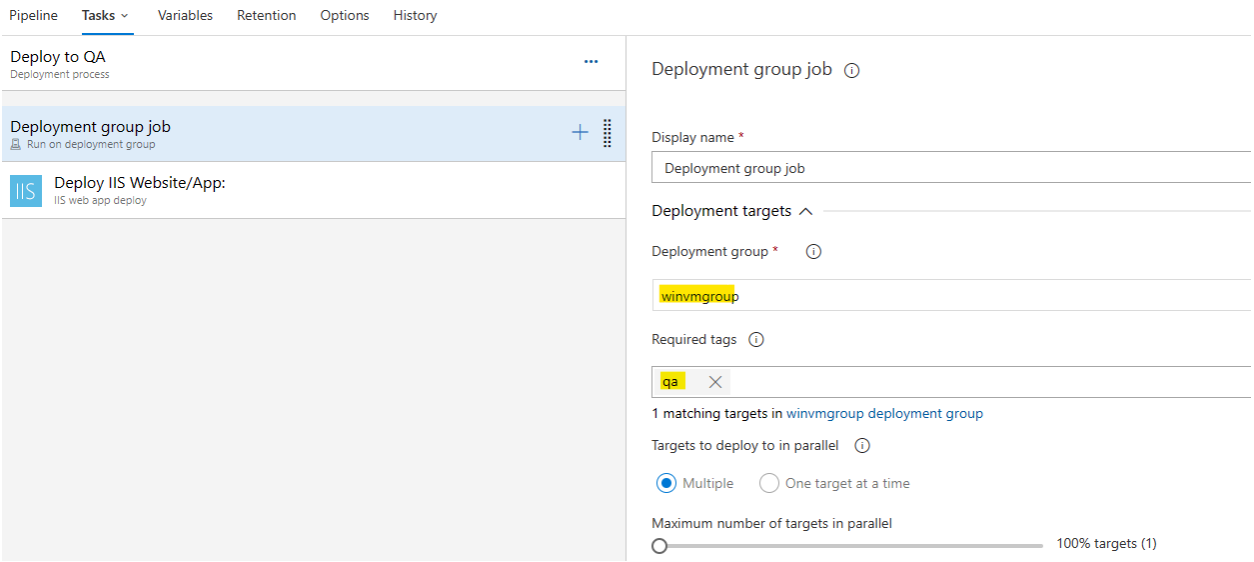


Artifacts: build pipeline 을 통해 빌드된 artifact (latest)지정

Stages: QA 와 Prod 용 두개의 stage 생성

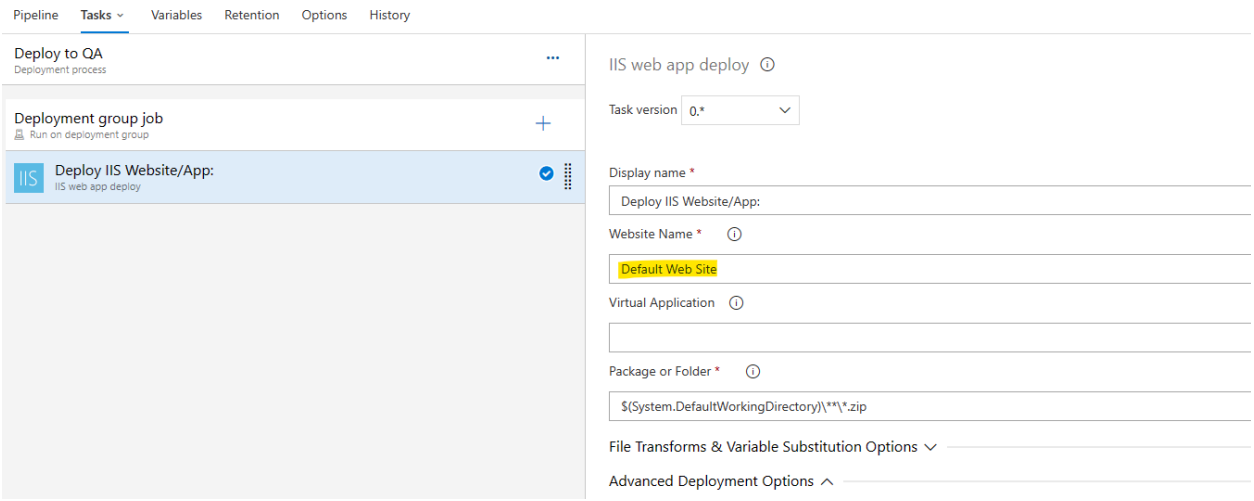
6.5 Config stage job

Stage Job 은 디폴트로 추가된 **Run on agent** 를 삭제하고, **Run on deployment group** 을 추가합니다. 설정 창에서 deployment group 과 Tag 를 지정합니다.



6.6 Edit task

IIS web app deploy Task 를 추가하고, 아래의 Website Name 에 **Default Web Site** 를 추가합니다.



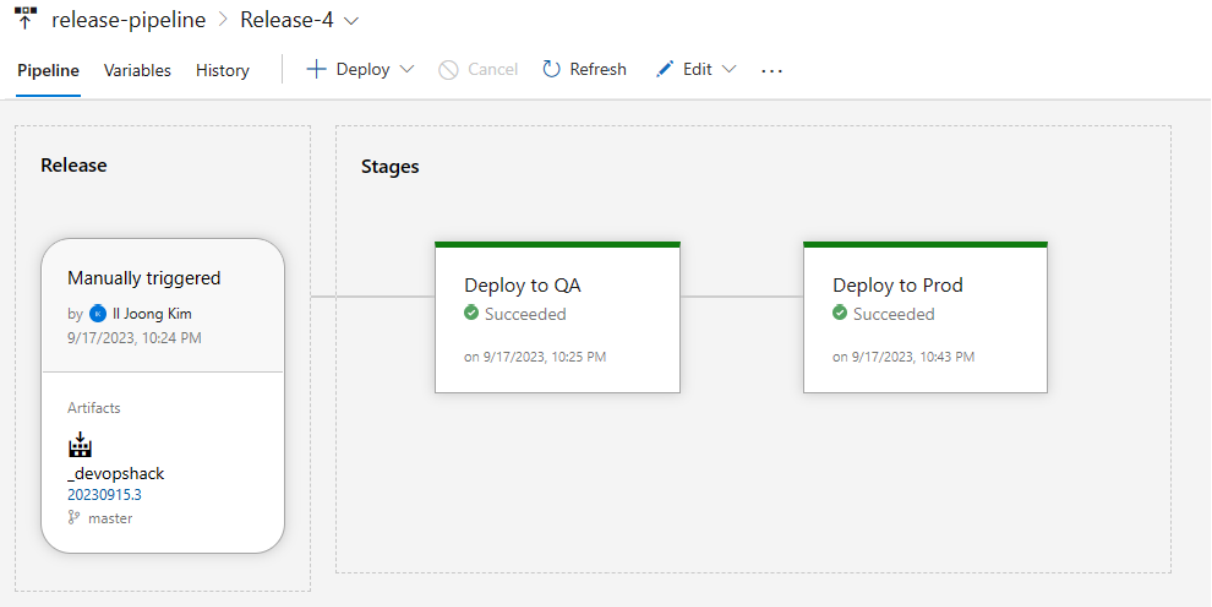
참고로 Build artifact(zip 파일)는 deployment group agent 를 통해 미리 local 에 복사된 후 Deploy IIS 와 같은 태스크가 실행되는 방식입니다.

Reference:

- Deploy to Azure VMs using deployment groups: https://docs.microsoft.com/en-us/azure/devops/pipelines/release/deployment-groups/deploying-azure-vms-deployment-groups?view=azure-devops

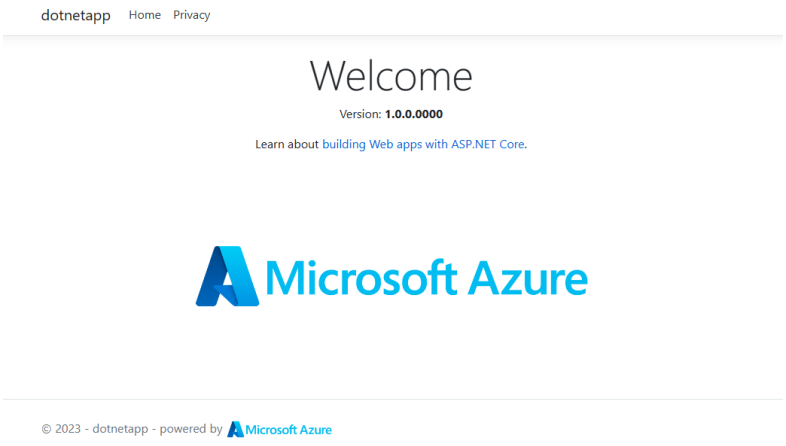
6.7 Trigger release

릴리즈를 트리거하여 배포를 진행합니다.



6.8 Test deployment

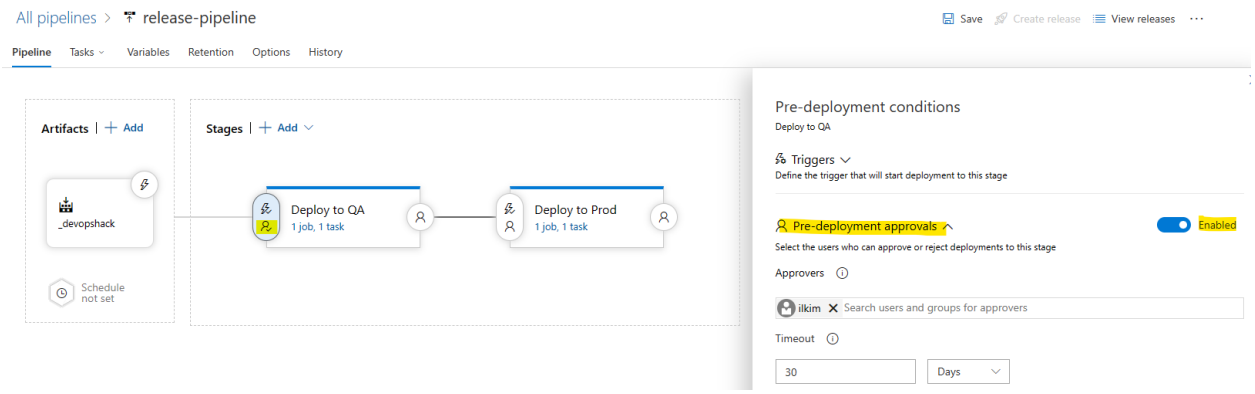
브라우저를 열고 해당 VM(QA 또는 Prod)의 IP 로 이동하여 배포 성공 여부 확인합니다.



7 Challenge 4: Approval, notification & branch policy

7.1 Approval

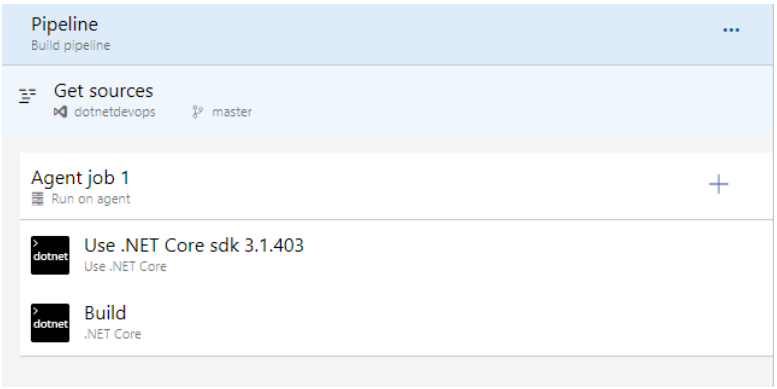
각 스테이지의 사전/사후 승인(approval)이 필요한 경우 아래와 같이 사람 아이콘(👤)을 클릭하여 승인자를 추가합니다.



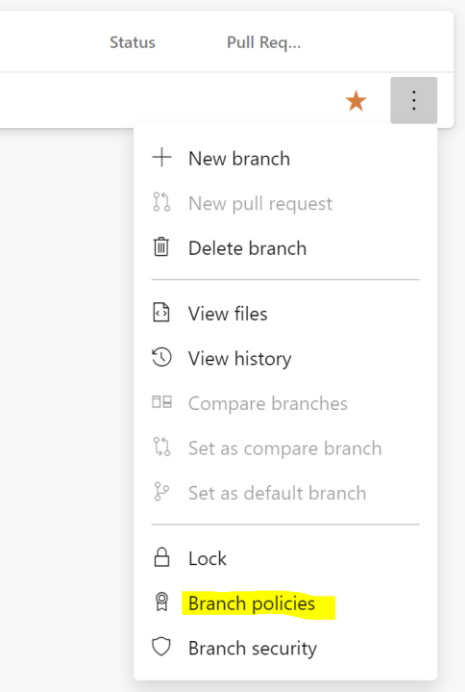
7.2 Branch policy

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

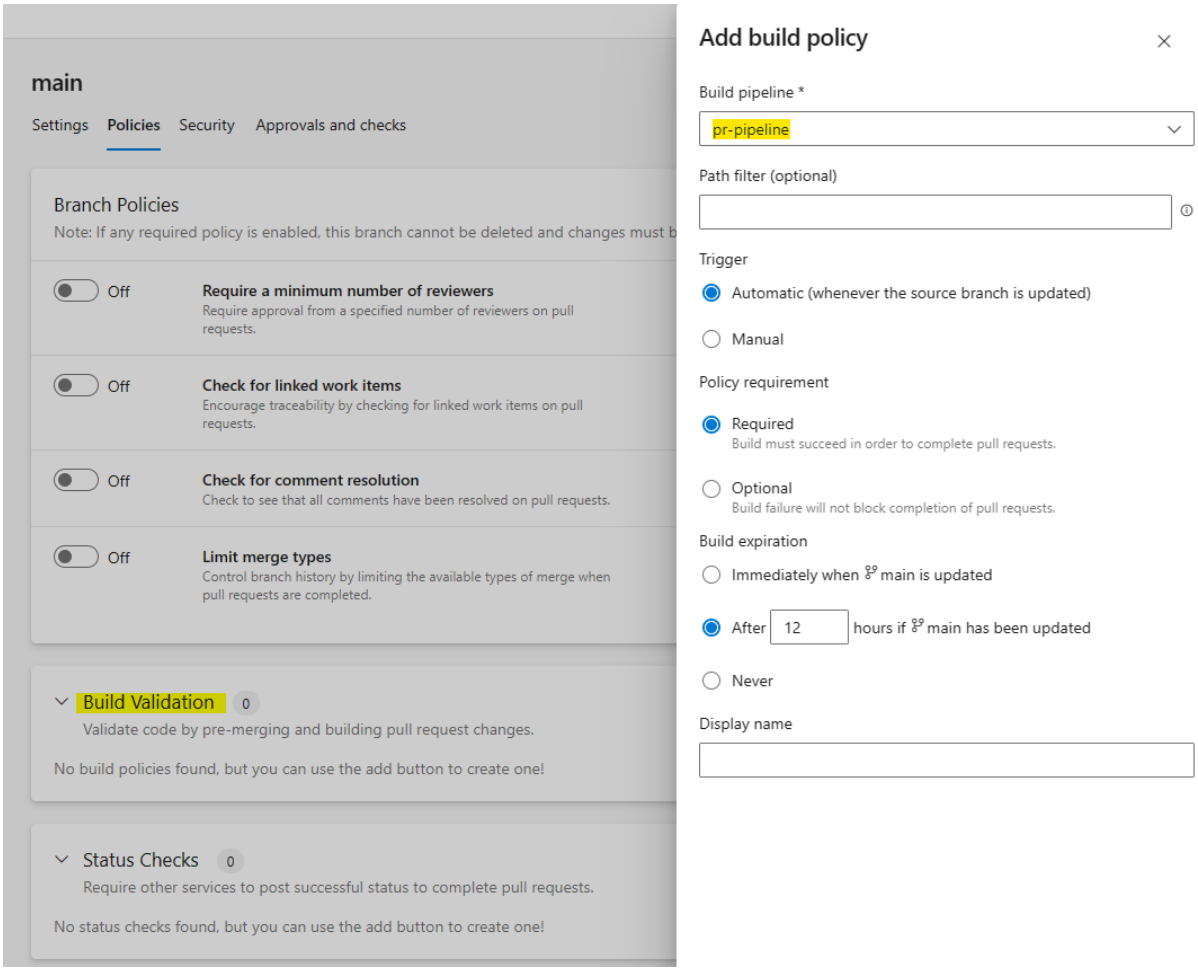
먼저, PR 빌드용 파이프라인을 생성하고 **pr-pipeline** 이름으로 지정합니다.



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

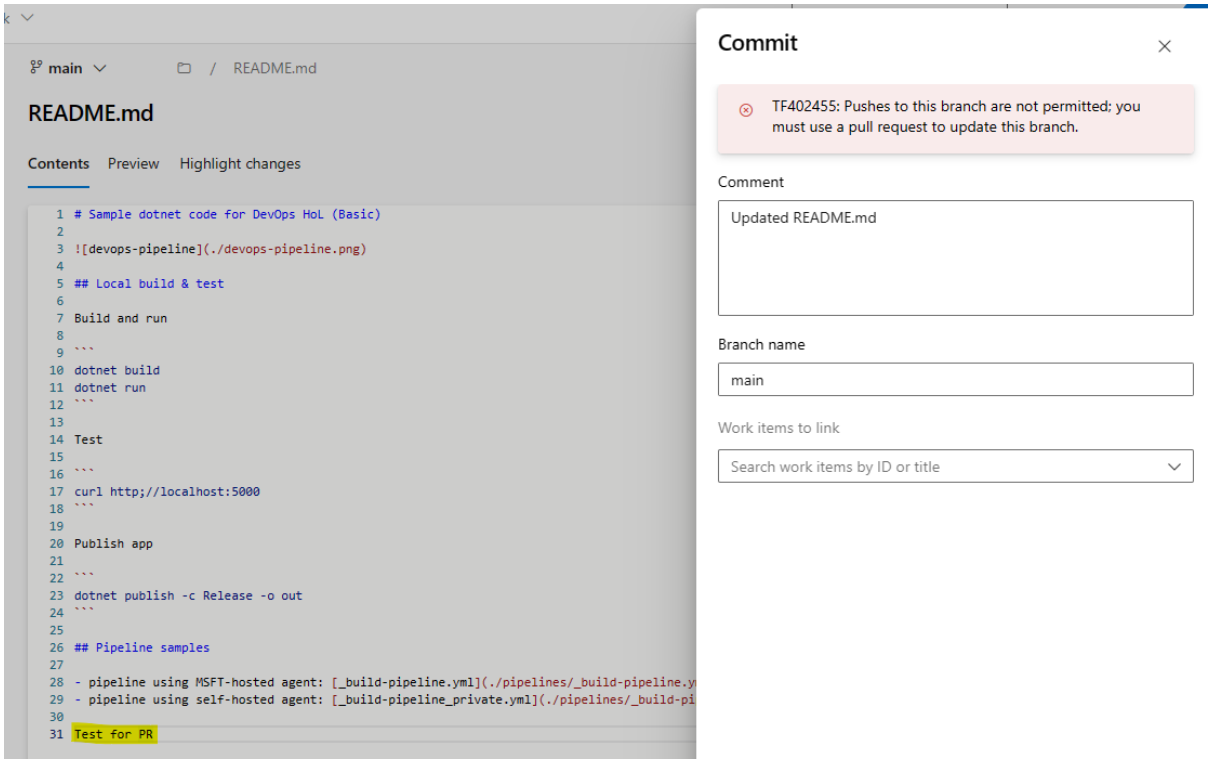


Build pipeline 항목에 앞서 생성한 pr-build-pipeline 를 선택합니다.



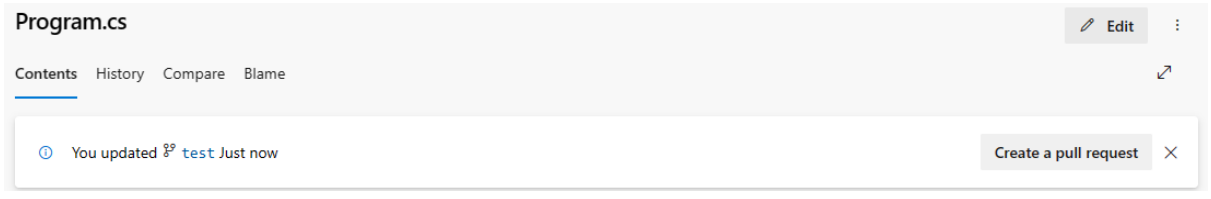
> **노트**: Path filter 에 제외(Exclude) 위치를 설정할 수 있음. 예를 들어 Build 성공여부가 필요 없는/pipelines, /docs 와 같은 path 변경 시에는 "`!/pipelines; !/docs`" 값을 추가하여 불필요한 Build 실행을 방지.

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

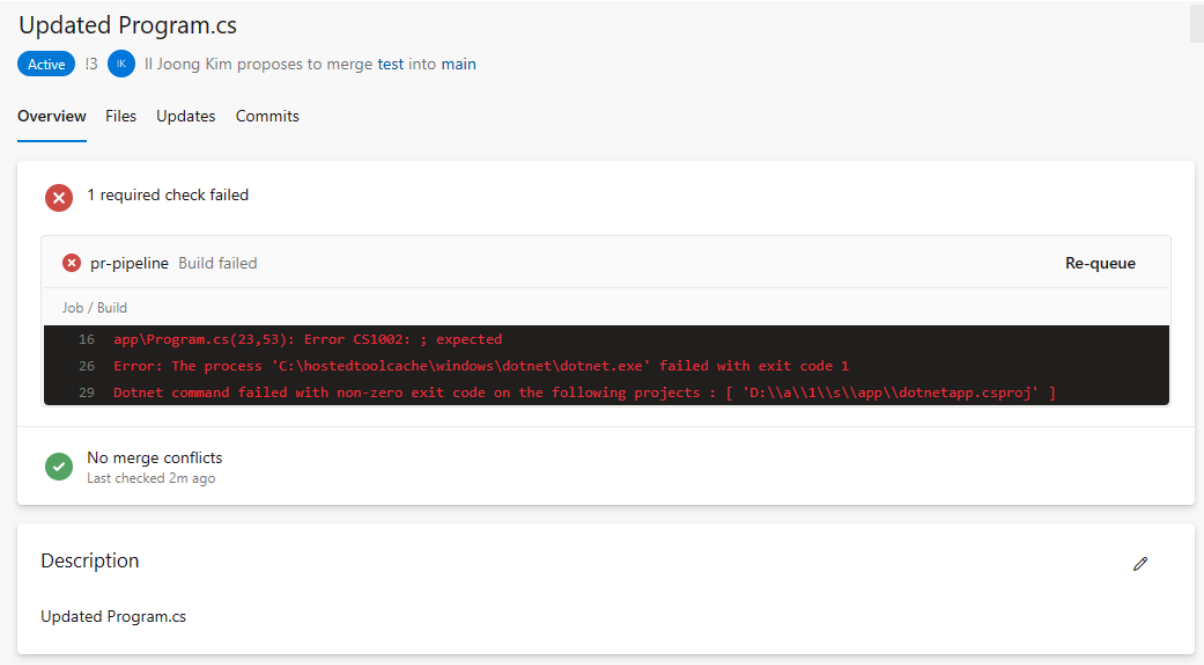


또한, 신규 브랜치(예: `test`)를 생성하고 Program.cs 가 오류가 발생하도록 수정(';'를 삭제)합니다.

```
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
```

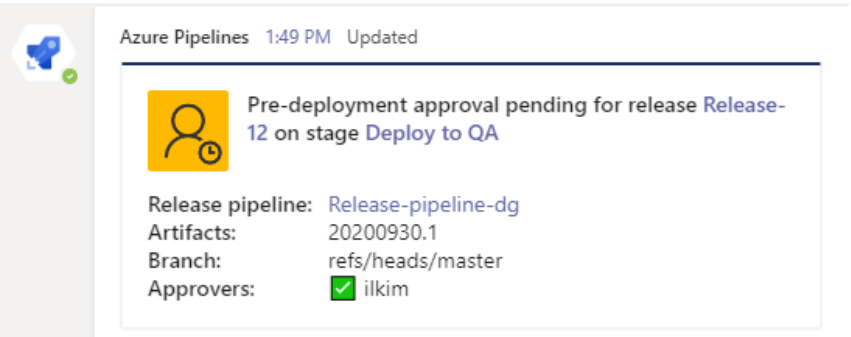


Commit 후, PR 을 요청하면 아래와 같이 오류가 발생한 것을 확인할 수 있습니다.



7.3 Teams notification

기본적으로 Approval 을 설정하며 메일로 approval 요청이 전달되고 Approval 은 Azure DevOps 로 이동하여 승인 또는 거절해야 합니다. Teams 또는 Slack 과 같은 메시지앱에 알림을 설정하면 메시지 알림에서 바로 승인/거절을 바로 할 수 있습니다. 아래의 화면은 Teams 에서 승인을 실행한 예입니다. 자세한 설정 방법은 Reference 를 참조하시기 바랍니다.



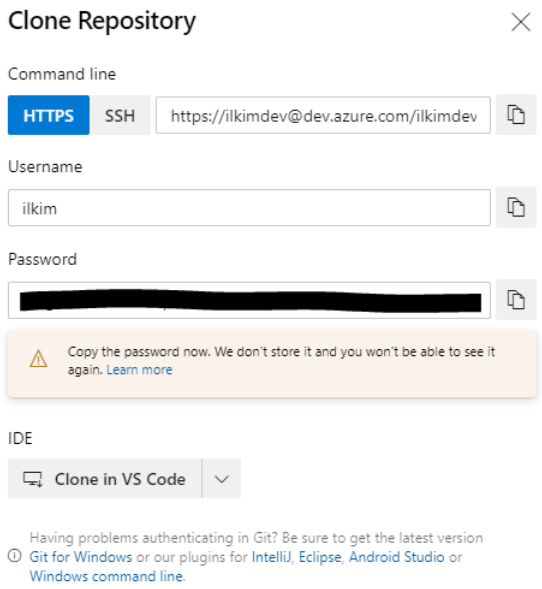
Reference:

https://learn.microsoft.com/en-us/azure/devops/pipelines/integrations/microsoft-teams?view=azure-devops

8 Appendix

8.1 Git clone/Credential

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



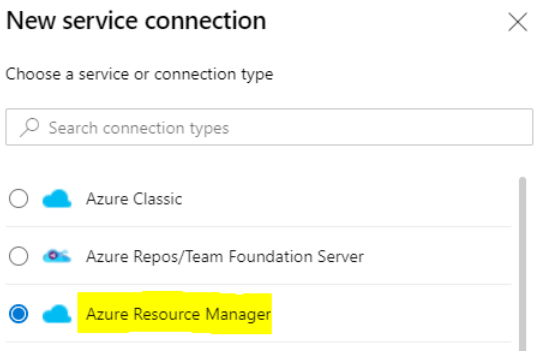
Git config 를 통해 credential 을 local 에 저장

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

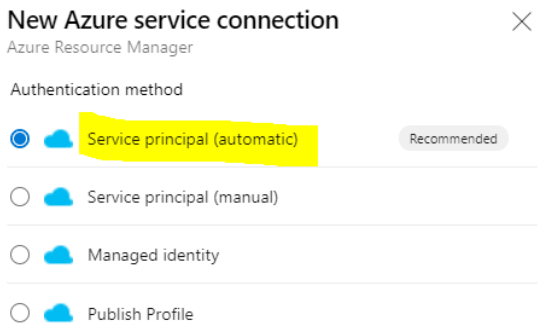
8.2 Service Connection

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

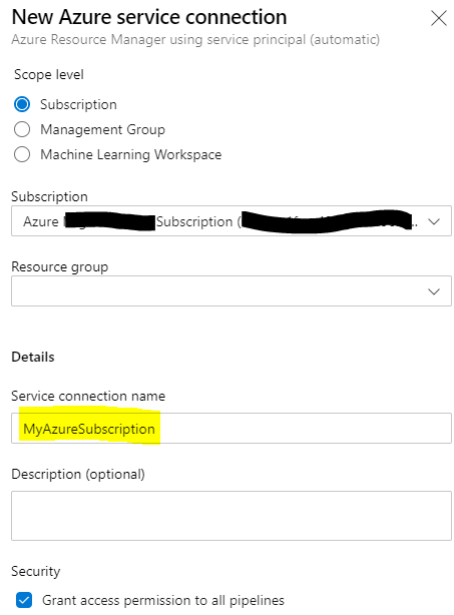
*Project settings->Service connections->***[Create/New service connection]**, *Azure Resource Manager* 선택하고 *[Next]*를 클릭합니다.



인증 방식은 추천 방식을 선택하고 **[Next]**를 클릭합니다.

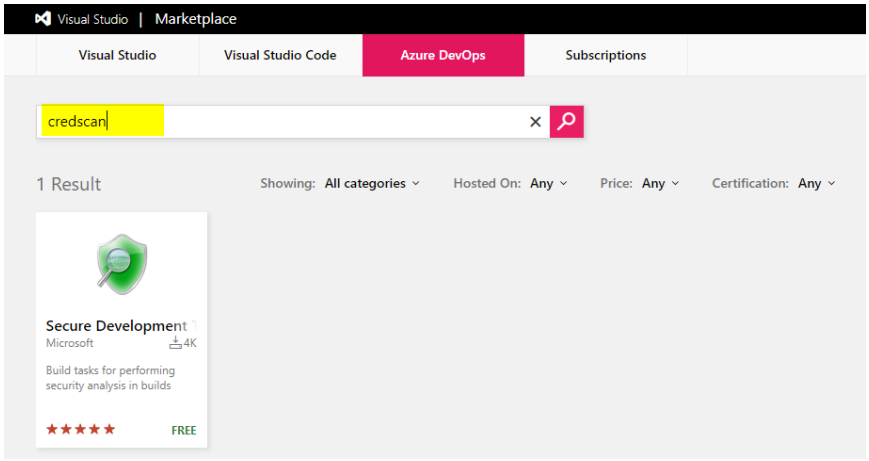


마지막으로, 연동할 구독을 선택하고 *service connection* 명을 지정하고 저장합니다.



8.3 Credscan

Marketplace 로 이동하여 credscan 을 검색하여 설치



설치 후 새로운 브라우저창에서 적용 가능