WorkshopPLUS - Essentials on Azure DevOps Services and GitHub

Lab Guides

Conditions and Terms of Use

Module 4: Azure Pipelines

Lab 1: Configuring CI/CD Pipelines as Code with YAML in Azure DevOps

Introduction

Azure Pipelines can be authored using a Classic Editor or with YAML files. Many teams prefer to define their builds using YAML (Yet Another Markup Language). This allows them to access the same build pipeline features as those using the classic editor, but with a markup file that can be managed like any other source file. YAML build definitions can be added to a project by simply adding their source file to the root of the repository.

Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

Objectives

In this lab, you will create a pipeline using YAML to build and deploy your code.

You will:

- Understand the basic features of YAML Pipelines.
- Understand the value of Pipelines as code.

YAML build definitions can be added to a project by simply adding their source file to the root of the repository.

Prerequisites

- Lab 2: PartsUnlimited Lab Setup

Estimated Time to Complete This Lab

30 minutes

Module 4: **Azure Pipelines**, **Lab 1: Configuring CI/CD Pipelines**, Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

---

Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

Objectives

Many teams prefer to define their builds using YAML (Yet Another Markup Language). This allows them to access the same build pipeline features as those using the classic editor, but with a markup file that can be managed like any other source file. YAML build definitions can be added to a project by simply adding their source file to the root of the repository.

Prerequisites

- Lab 2: PartsUnlimited Lab Setup

Tasks

Module 4: **Azure Pipelines**, **Lab 1: Configuring CI/CD Pipelines**, Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

---

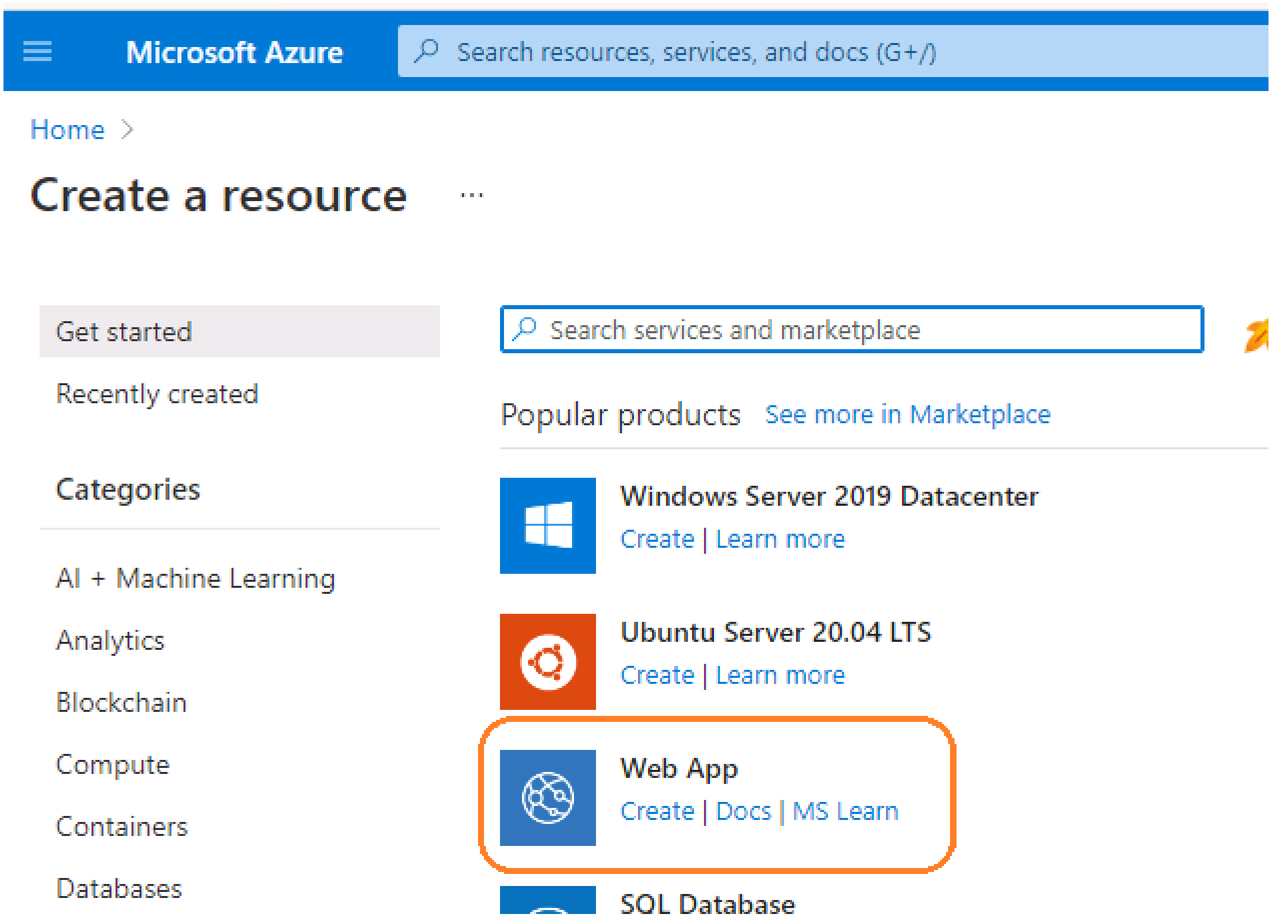
Task 1: Creating Azure Resources

1. This lab requires a deployment of a sample ASP.NET Core application to an Azure App Service. To do this, you will need to spin up the necessary resources. Open a new browser tab and navigate to the **Azure Portal**: **https://portal.azure.com**

2. When asked for the **username**, enter the same username and password that you used for logging into Azure DevOps Services.

3. Click on **Create a resource** and click **Web App**.

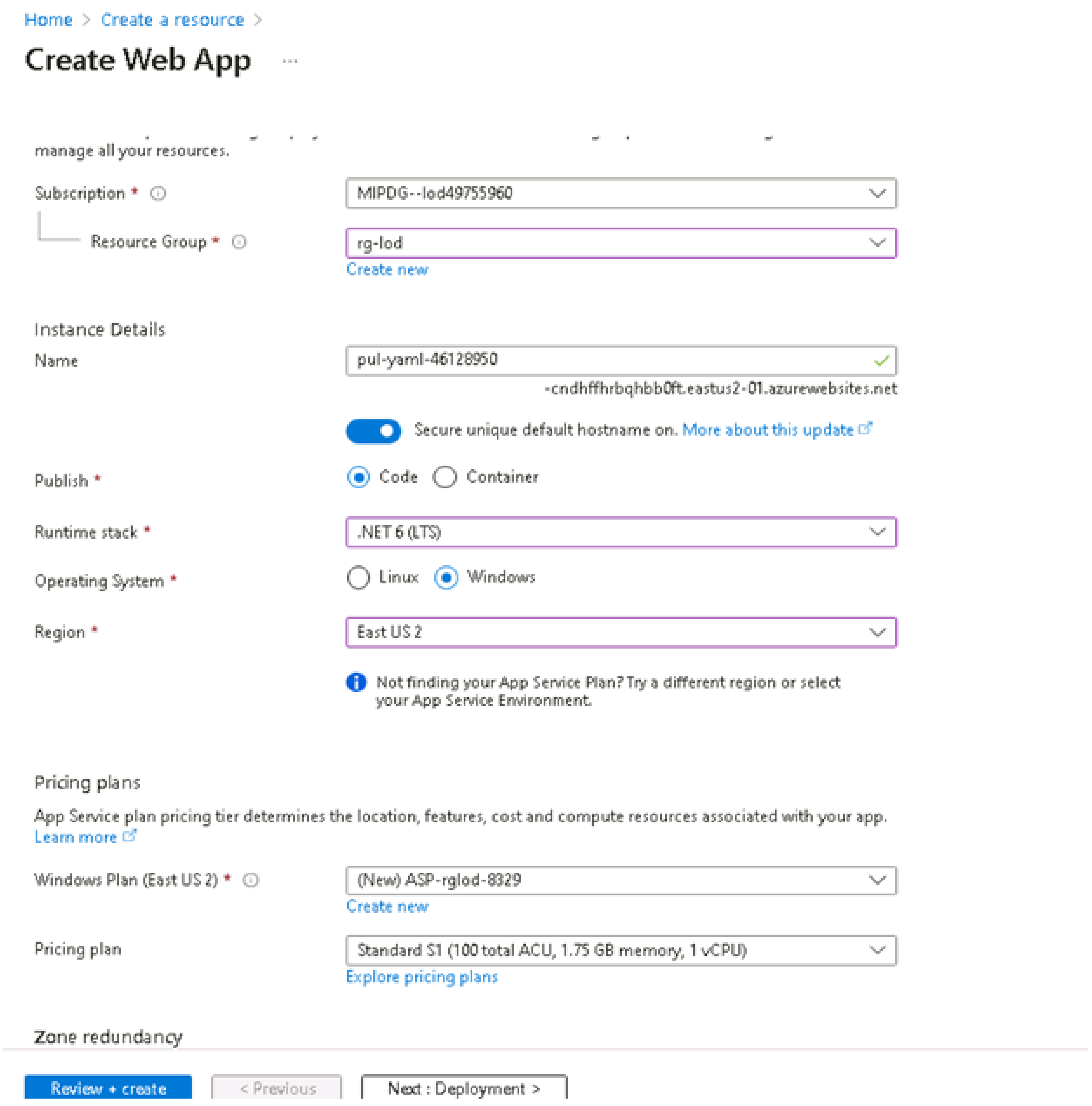
   > If **Web App** is not shown in the list then search for it in the **Search services and marketplace**

4. Enter following details in the **Create Web App** page:

   - Resource Group: **rg-lod**
   - Name: **pul-yaml-[YourInitials]** (Make sure the name is unique across all of Azure)
   - Publish: **Code**
   - Runtime stack: **.NET 6 (LTS)**
   - Operating System: **Windows**
   - Region: **East US 2**
   - Windows Plan: **Leave the default name selected**

- Pricing plan: **Standard S1**
- Click **Review + create** and then **Create**.



manage all your resources.

Subscription *: MIPDG--lod49755960

Resource Group *: rg-lod
Create new

Instance Details

Name: pul-yaml-46128950
-cndhffhrbqhbb0ft.eastus2-01.azurewebsites.net

Secure unique default hostname on. More about this update

Publish *: [✓ Code] Container

Runtime stack *: .NET 6 (LTS)

Operating System *: Linux [✓ Windows]

Region *: East US 2

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

Windows Plan (East US 2) *: (New) ASP-rglod-8329
Create new

Pricing plan: Standard S1 (100 total ACU, 1.75 GB memory, 1 vCPU)
Explore pricing plans

Zone redundancy

> If you encounter issues while creating the Web App, feel free to change the **Region** of the Web App (e.g Central US) and retry.

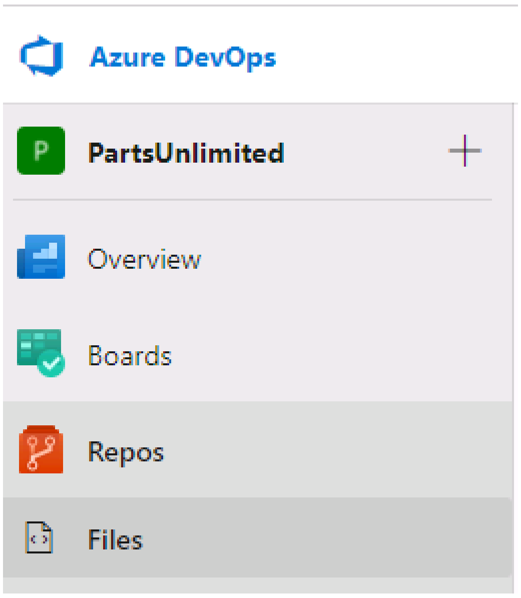
Module 4: **Azure Pipelines**, **Lab 1: Configuring CI/CD Pipelines**, Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

---

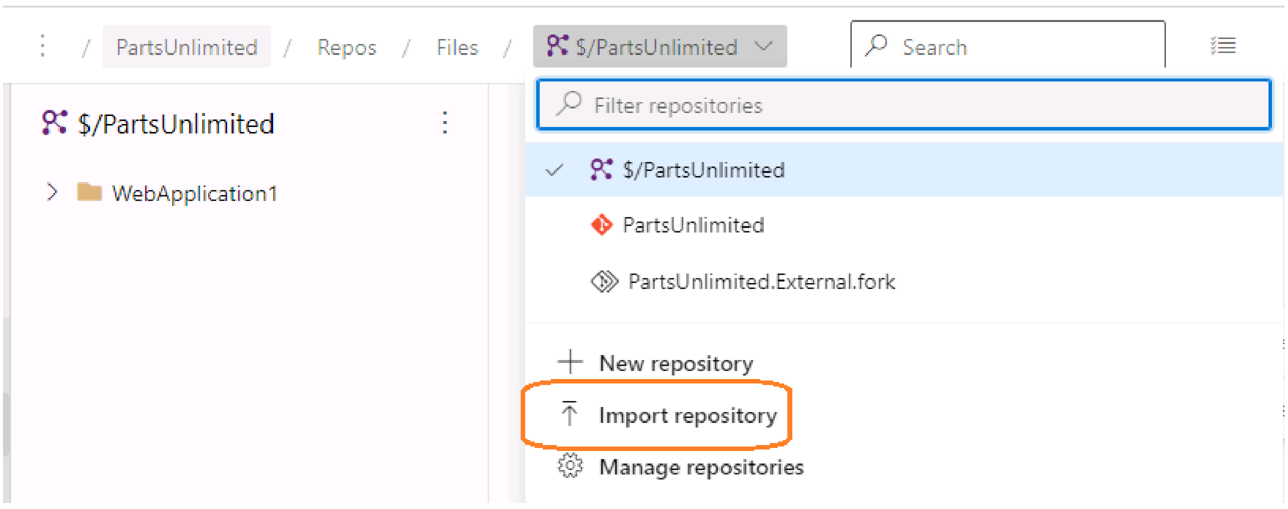
Task 2: Configuring the PartsUnlimited Project

1. Switch back to **Azure DevOps Services**

2. Navigate to **Repos | Files** for the **PartsUnlimited** project.

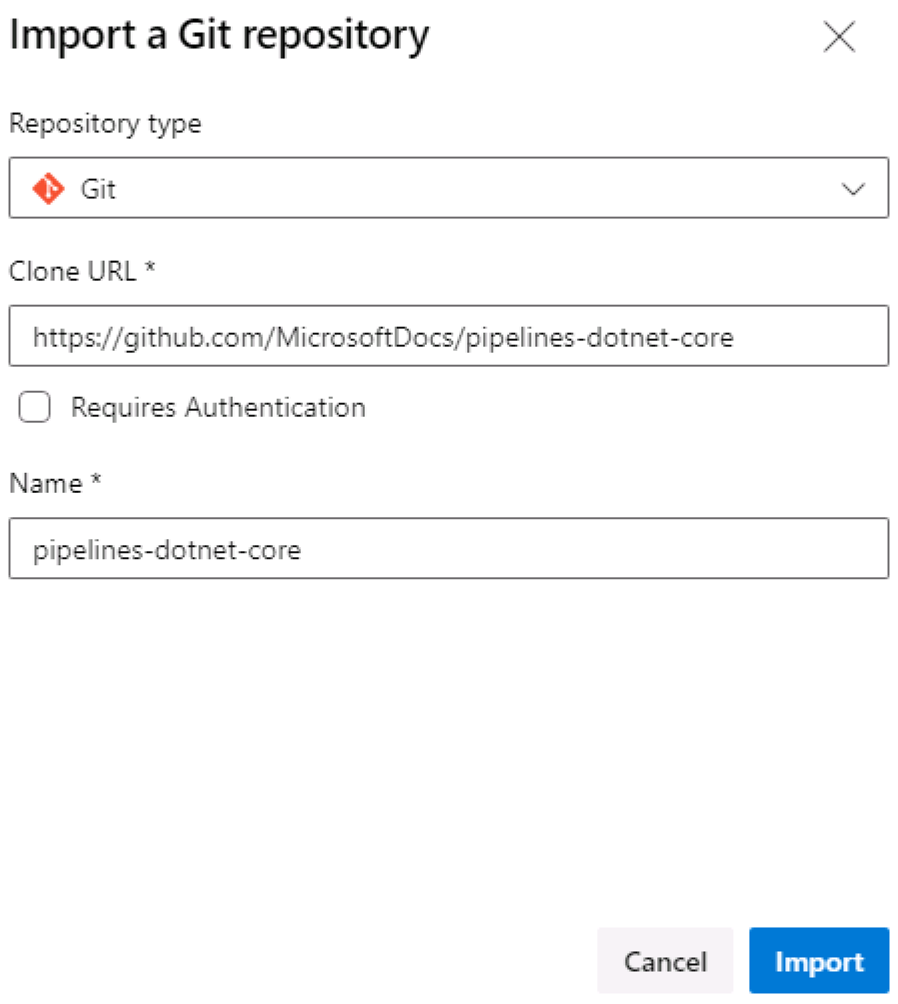


3. From the top-center dropdown, select **Import repository**.



4. In the **Import a Git repository** page, enter following information:

   - Repository type: **Git**
   - Clone URL: **https://github.com/MicrosoftDocs/pipelines-dotnet-core**
   - Name: **pipelines-dotnet-core** (choose the default name populated here)
   - Click **Import**



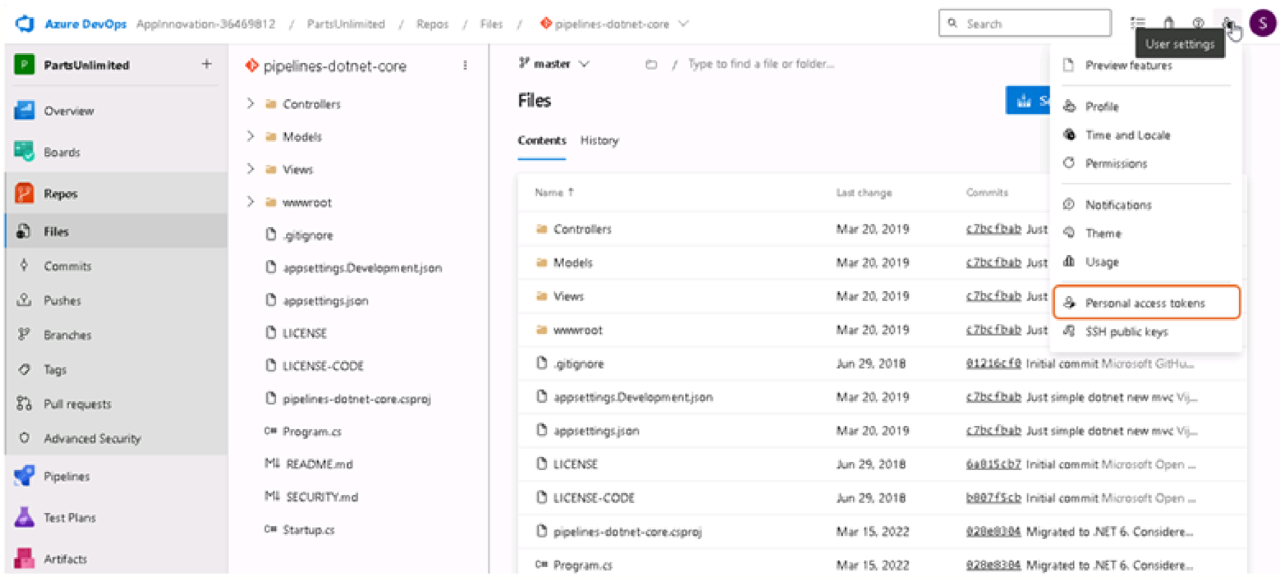
> After a few seconds you will see an **Import Successful** message and you will be redirected to this newly created repository in Azure Repos.

Module 4: **Azure Pipelines**, **Lab 1: Configuring CI/CD Pipelines**, Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

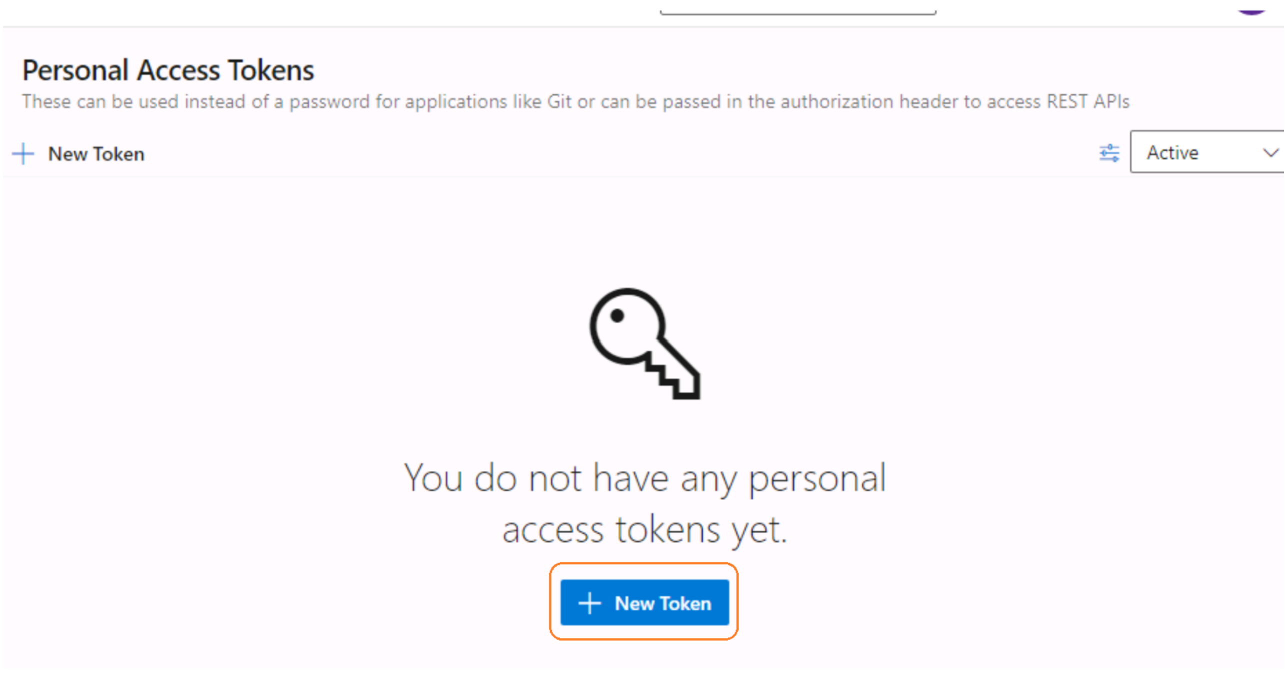
---

Task 3: Configuring a self-hosted agent

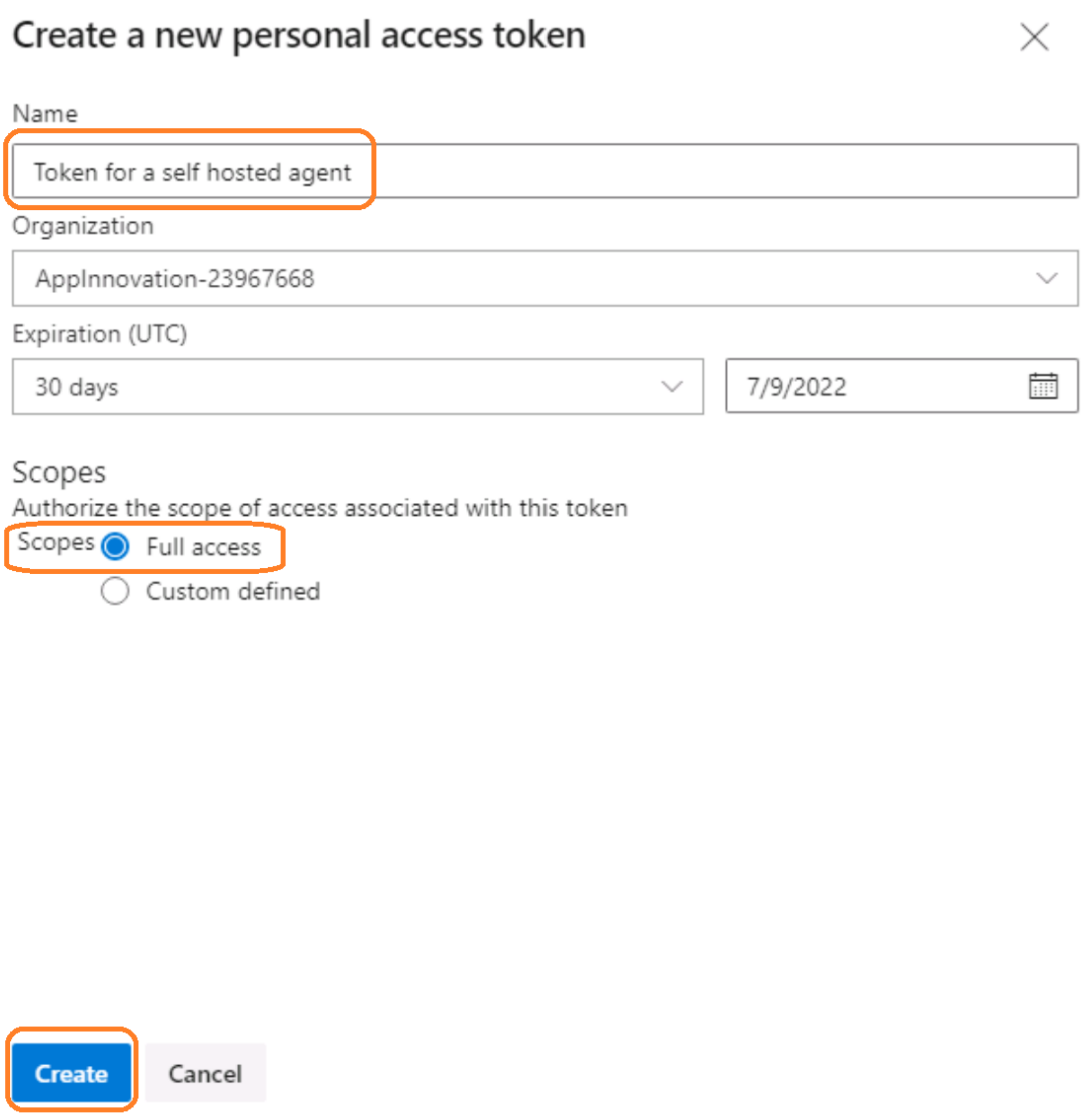
1. Click on **User settings** and select **Personal access tokens**.



2. Click on **+ New Token**.


Personal Access Tokens
These can be used instead of a password for applications like Git or can be passed in the authorization header to access REST APIs
New Token
Active
You do not have any personal access tokens yet.
New Token

3. Enter the name of the token as **Token for a self hosted agent**, Scopes as **Full access** and click **Create**.


Create a new personal access token

Name: Token for a self hosted agent

Organization: AppInnovation-23967668

Expiration (UTC): 30 days 7/9/2022

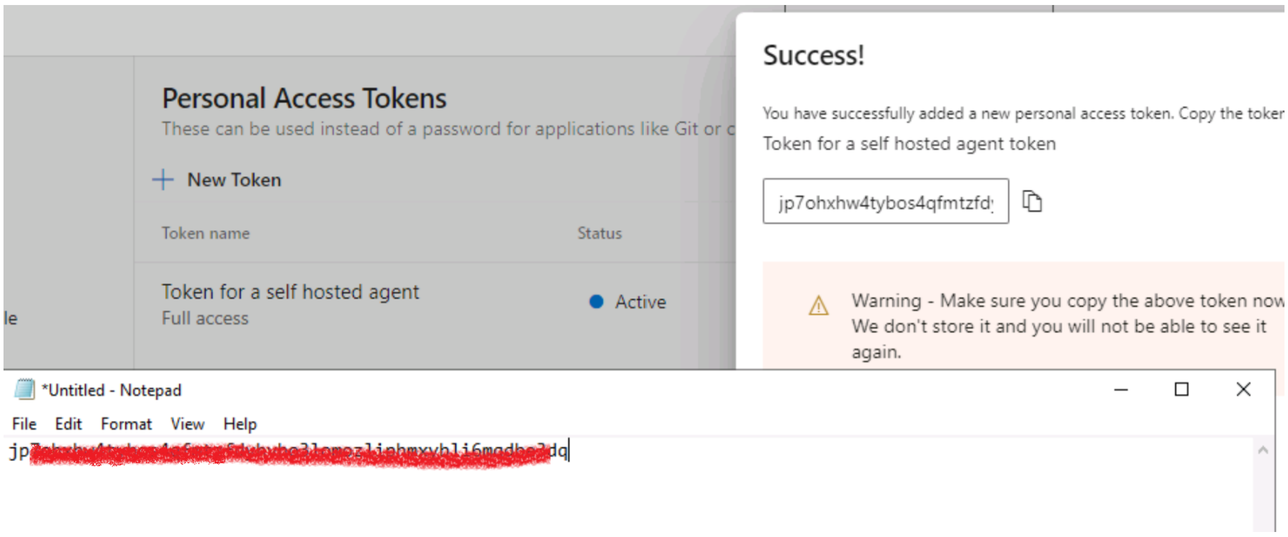
Scopes

Authorize the scope of access associated with this token

Scopes [✓ Full access] Custom defined

> Typically, you wouldn't create a token with the full access. However, for the purpose of this lab, we will create a token with the full access.

4. As the token gets generated, copy the token and paste it in a notepad file so that you can use it later.



Success!

You have successfully added a new personal access token. Copy the toker

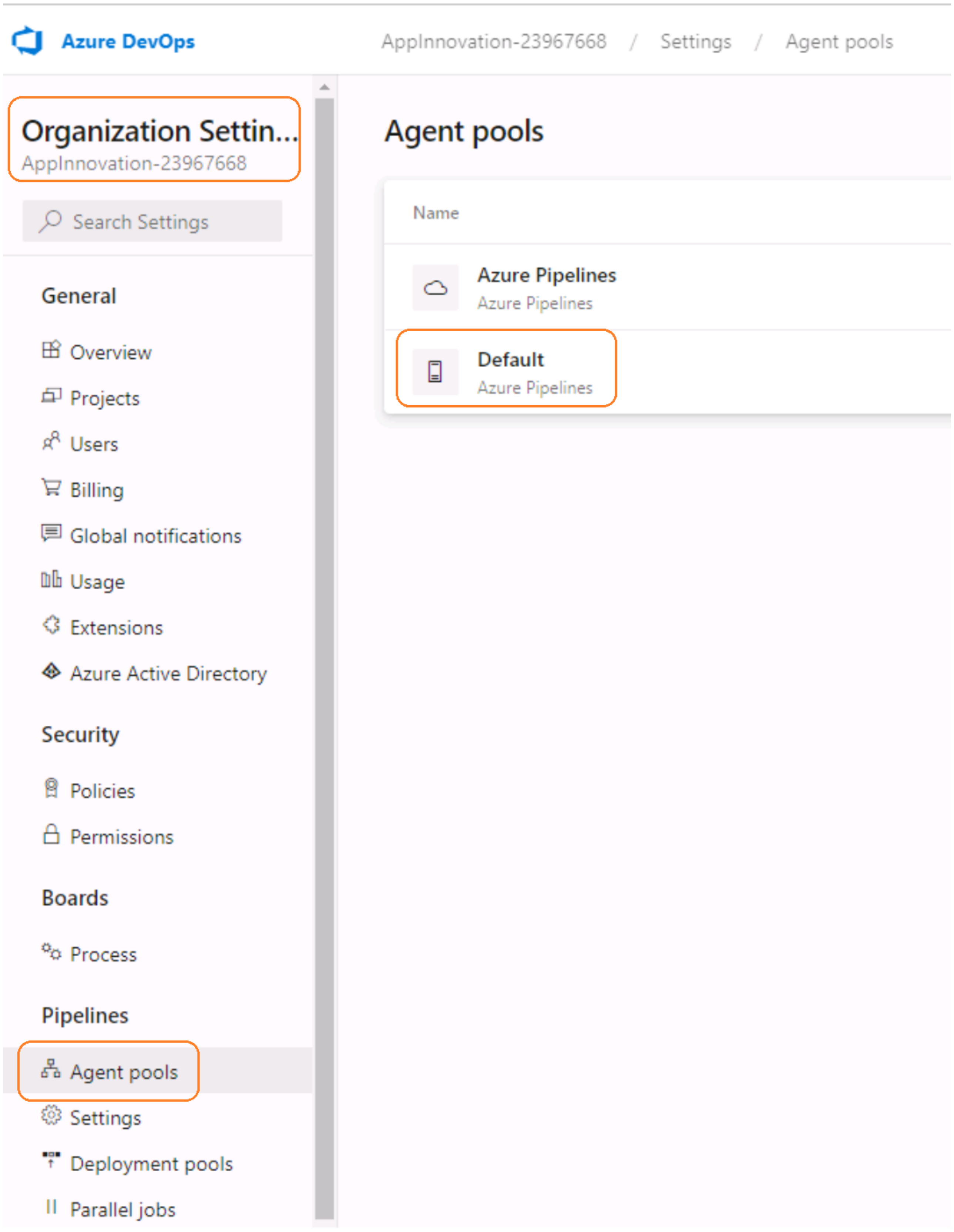
Token for a self hosted agent token

jp7ohxhw4tybos4qfmtzfd

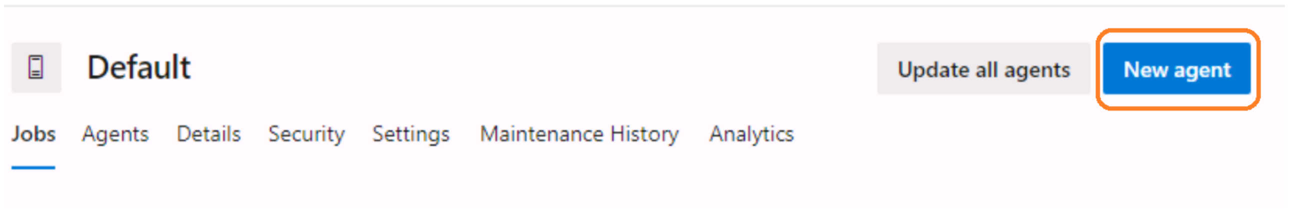
Warning - Make sure you copy the above token now. We don't store it and you will not be able to see it again.

```
jp[redacted]dq
```

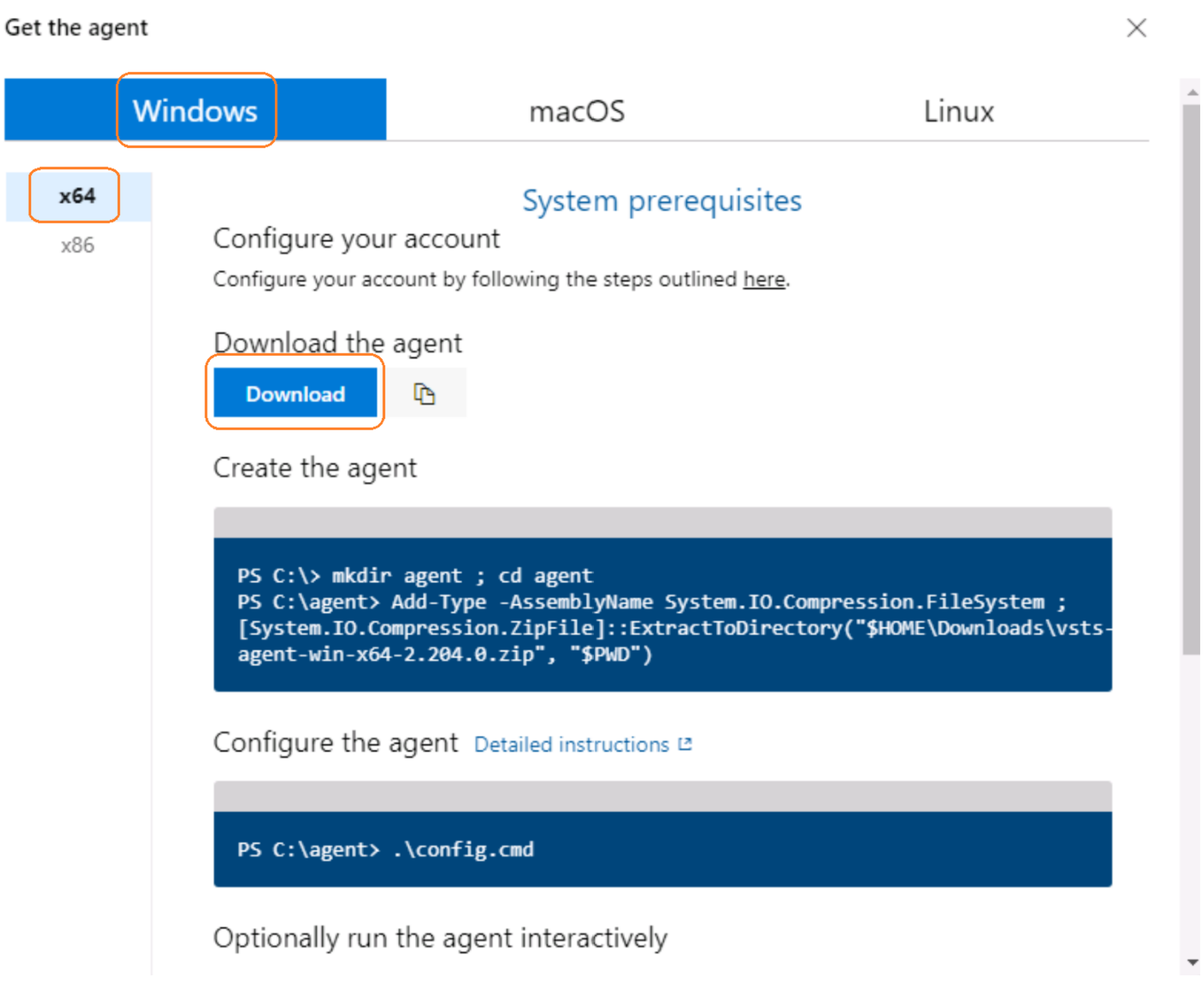
5. **Close** the token window and navigate to the **Organization Settings** and select **Agent pools**. Click on **Default**.



6. Click on **Agents** and select **New Agent**.



7. Select **Windows**, **x64** and **Download** the agent.



Configure your account

Configure your account by following the steps outlined here.

Download the agent

Create the agent

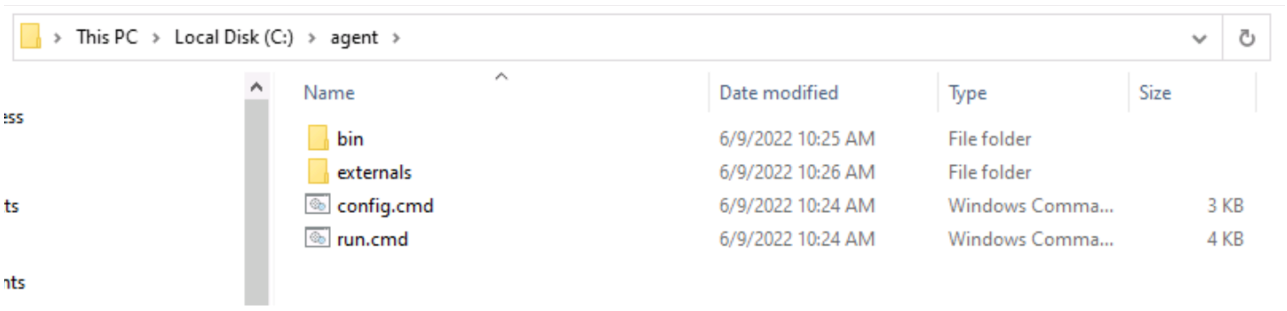
```
PS C:\> mkdir agent ; cd agent
PS C:\agent> Add-Type -AssemblyName System.IO.Compression.FileSystem ;
[System.IO.Compression.ZipFile]::ExtractToDirectory("$HOME\Downloads\vsts-
agent-win-x64-2.204.0.zip", "$PWD")
```

Configure the agent

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

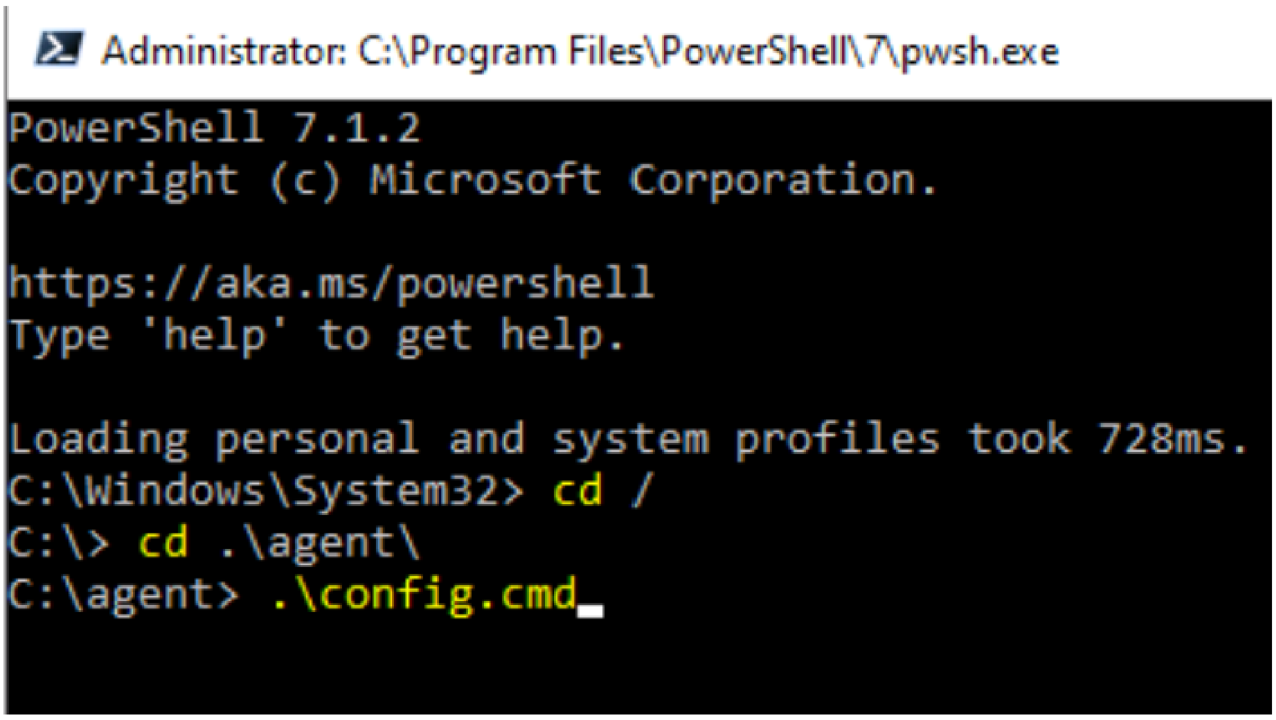
Optionally run the agent interactively

8. **Create** a folder named "agent" under the C: drive **(C:\agent)**. Then **unzip** the downloaded file and **copy** the contents under C:\agent folder.



Name	Date modified	Type	Size
bin	6/9/2022 10:25 AM	File folder	
externals	6/9/2022 10:26 AM	File folder	
config.cmd	6/9/2022 10:24 AM	Windows Comma...	3 KB
run.cmd	6/9/2022 10:24 AM	Windows Comma...	4 KB

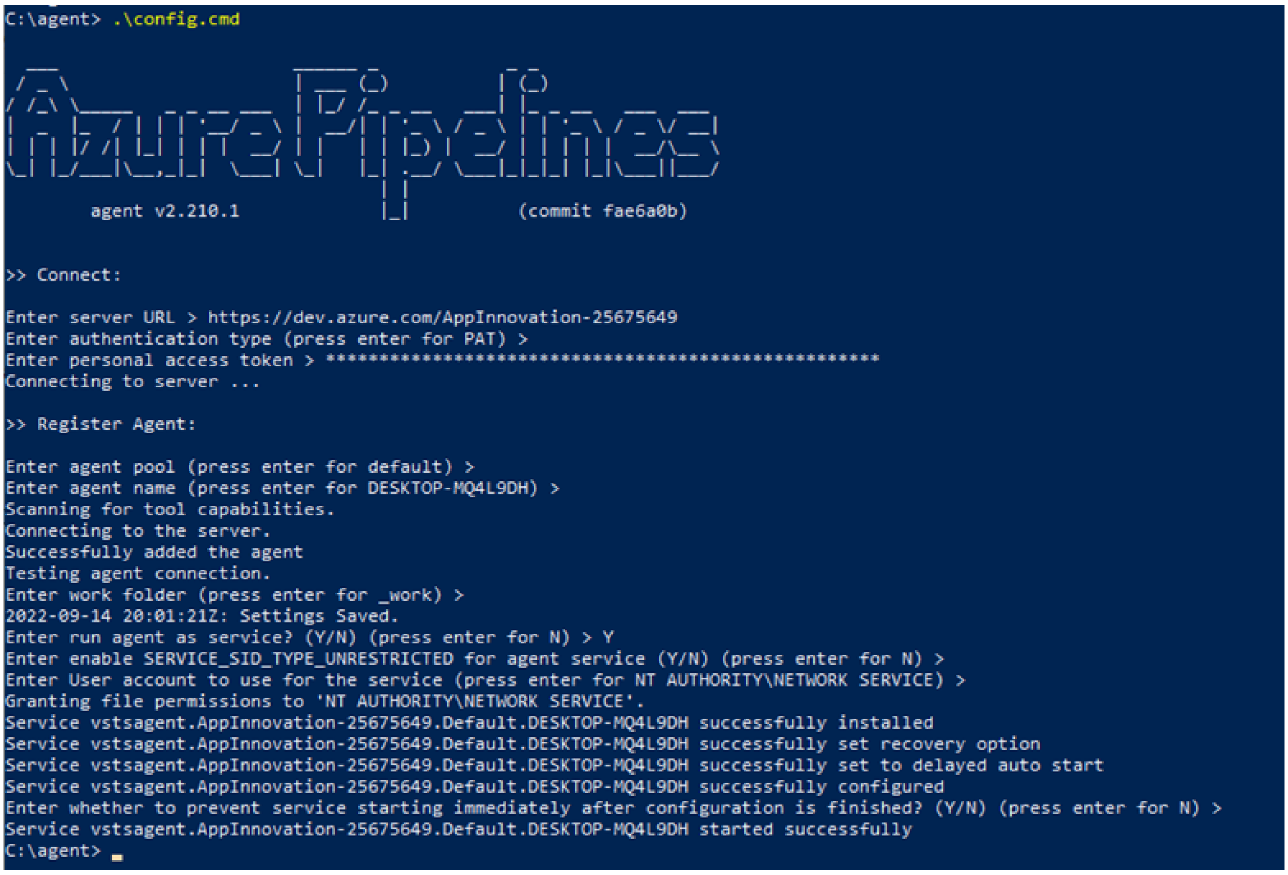
9. Run **PowerShell as Administrator** and navigate to **c:\agent** folder. Run **.\config.cmd** command.



```
PowerShell 7.1.2
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

Loading personal and system profiles took 728ms.
C:\Windows\System32> cd /
C:\> cd .\agent\
C:\agent> .\config.cmd
```

10. Enter following details for configuring the self-hosted agent:
    - *Enter server URL* >: **https://dev.azure.com/AppInnovation-[YourName]**
    - *Enter authentication type (press enter for PAT)* >: **Press Enter to select PAT**
    - *Enter personal access token* >: **Paste the PAT you copied and pasted in the notepad file earlier**
    - *Enter agent pool (press enter for default)* >: **Press Enter to select Default agent pool**
    - *Enter agent name (press enter for DESKTOP-MQ4L9DH)* >: **Press Enter to select the default agent name**
    - *Enter work folder (press enter for _work)* >: **Press Enter**
    - *Enter run agent as service? (Y/N) (press enter for N)* >: Enter **Y** and press Enter
    - *Enter enable SERVICE_SID_TYPE_UNRESTRICTED for agent service (Y/N) (press enter for N)* >: **Press Enter**
    - *Enter User account to use for the service (press enter for NT AUTHORITY\NETWORK SERVICE)* >: **Press Enter to select Network Service**
    - *Enter whether to prevent service starting immediately after configuration is finished? (Y/N) (press enter for N)* >: **Press Enter to select No to start the agent immediately**

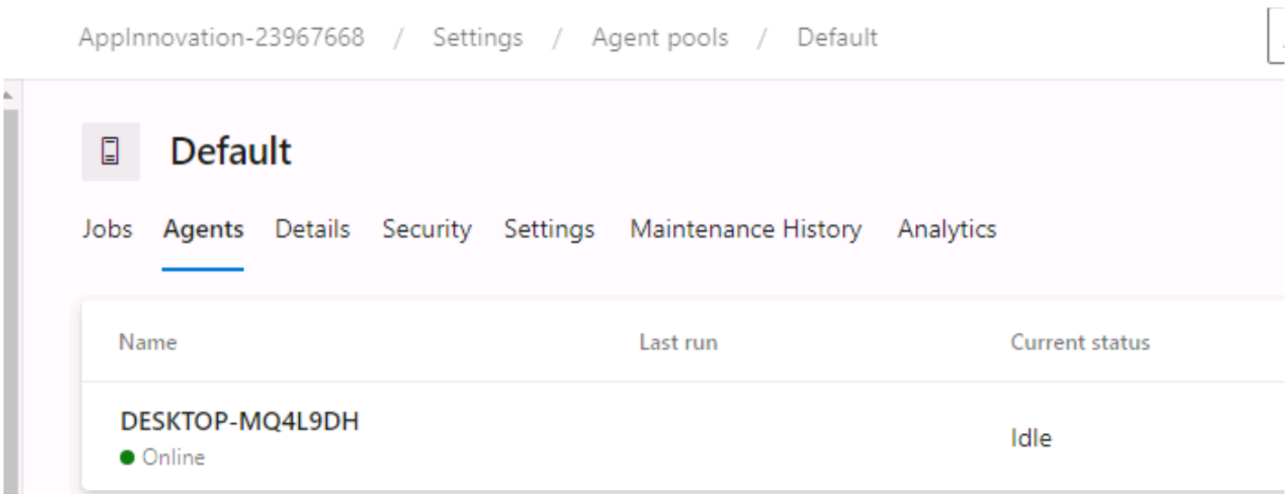
    This should configure and start the agent successfully.



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

        agent v2.210.1                          (commit fae6a0b)

>> Connect:

Enter server URL > https://dev.azure.com/AppInnovation-25675649
Enter authentication type (press enter for PAT) >
Enter personal access token > ****************************************************
Connecting to server ...

>> Register Agent:

Enter agent pool (press enter for default) >
Enter agent name (press enter for DESKTOP-MQ4L9DH) >
Scanning for tool capabilities.
Connecting to the server.
Successfully added the agent
Testing agent connection.
Enter work folder (press enter for _work) >
2022-09-14 20:01:21Z: Settings Saved.
Enter run agent as service? (Y/N) (press enter for N) > Y
Enter enable SERVICE_SID_TYPE_UNRESTRICTED for agent service (Y/N) (press enter for N) >
Enter User account to use for the service (press enter for NT AUTHORITY\NETWORK SERVICE) >
Granting file permissions to 'NT AUTHORITY\NETWORK SERVICE'.
Service vstsagent.AppInnovation-25675649.Default.DESKTOP-MQ4L9DH successfully installed
Service vstsagent.AppInnovation-25675649.Default.DESKTOP-MQ4L9DH successfully set recovery option
Service vstsagent.AppInnovation-25675649.Default.DESKTOP-MQ4L9DH successfully set to delayed auto start
Service vstsagent.AppInnovation-25675649.Default.DESKTOP-MQ4L9DH successfully configured
Enter whether to prevent service starting immediately after configuration is finished? (Y/N) (press enter for N) >
Service vstsagent.AppInnovation-25675649.Default.DESKTOP-MQ4L9DH started successfully
C:\agent>
```

11. Switch back to the browser to navigate to the **Agent Pools** in the **Organization Settings** and select **Agents**. You will see the agent *Online*.

> You might need to refresh the browser to see the agent as **Online**



Name	Last run	Current status
DESKTOP-MQ4L9DH ● Online		Idle

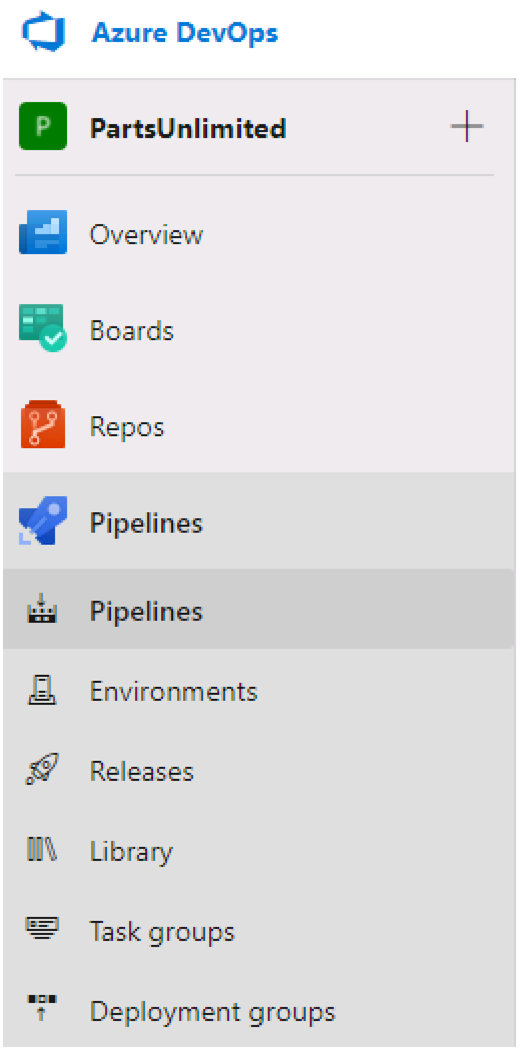
> It is also possible to run your Azure Pipelines agent in Docker. You can set up a self-hosted agent in Azure Pipelines to run inside a Windows Server Core (for Windows hosts), or Ubuntu container (for Linux hosts) with Docker. This is useful when you want to run agents with outer orchestration, such as Azure Container Instances.

Module 4: **Azure Pipelines**, **Lab 1: Configuring CI/CD Pipelines**, Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

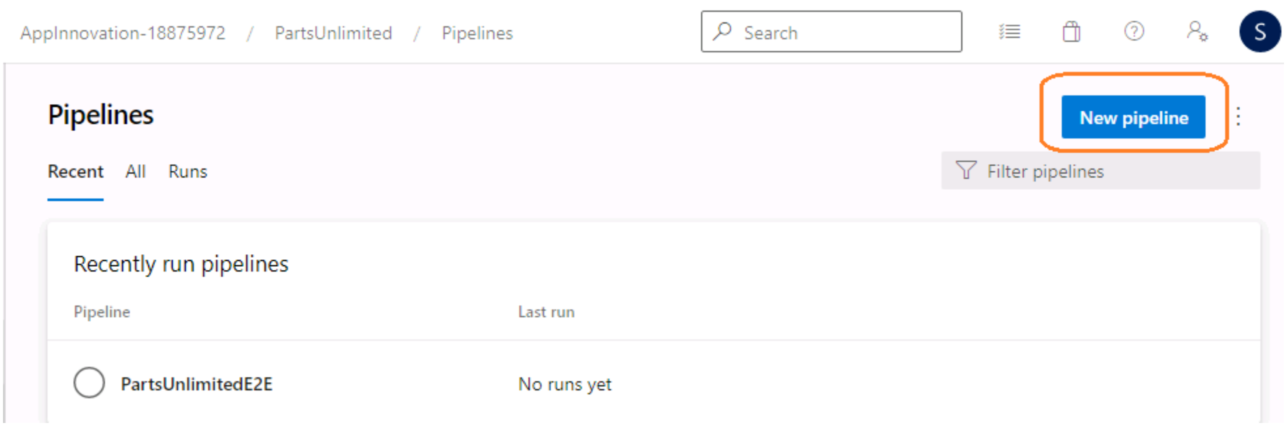
---

Task 4: Adding a YAML definition

1. Navigate to the **Pipelines** hub in Azure DevOps while staying inside the **PartsUnlimited** project.



2. Click **New pipeline** from the top-right corner.

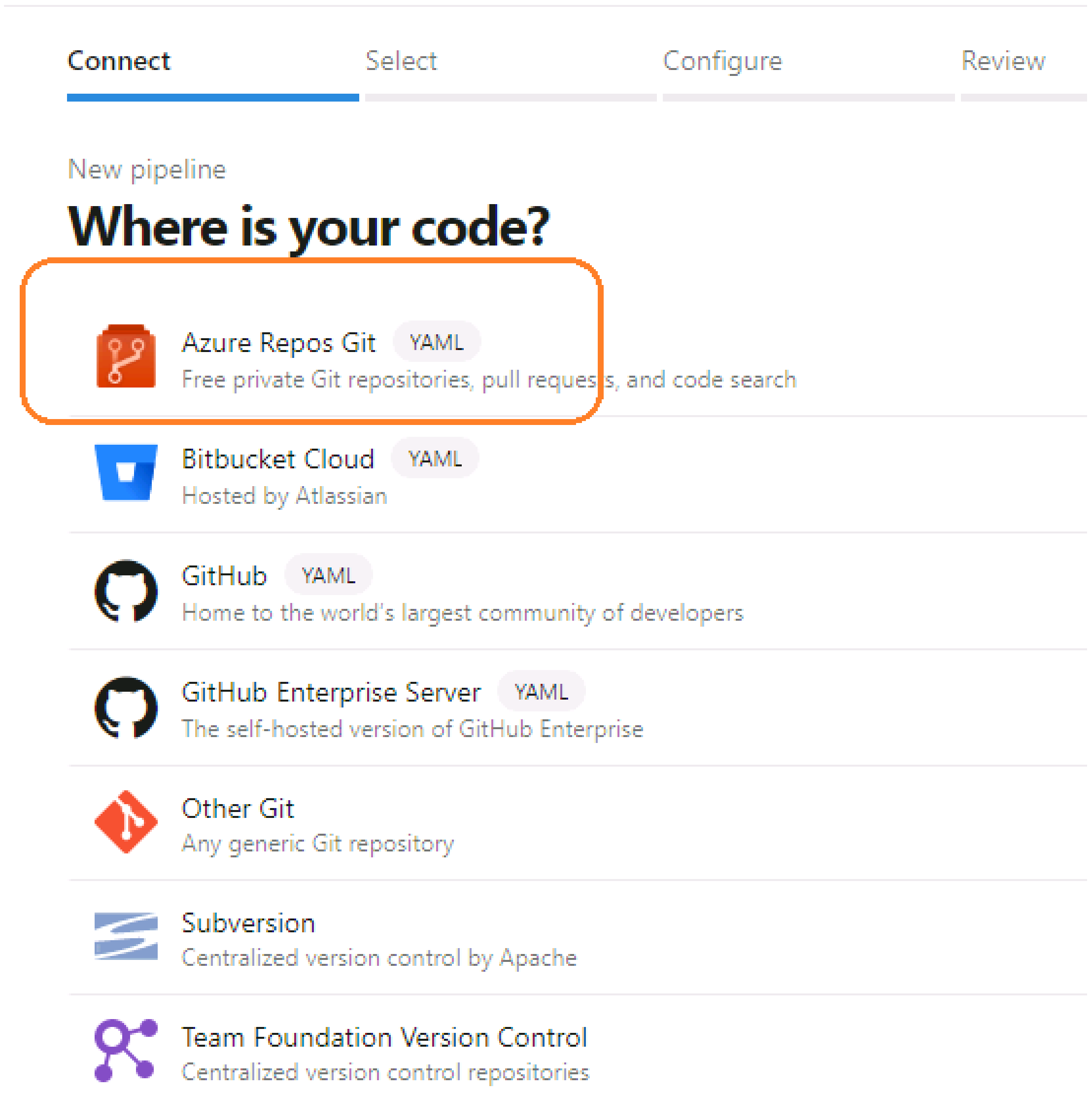


Recently run pipelines

Pipeline	Last run
PartsUnlimitedE2E	No runs yet

> We will use the wizard to automatically create the YAML definition based on our project.

3. Select the **Azure Repos Git** as the source hosting platform.



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

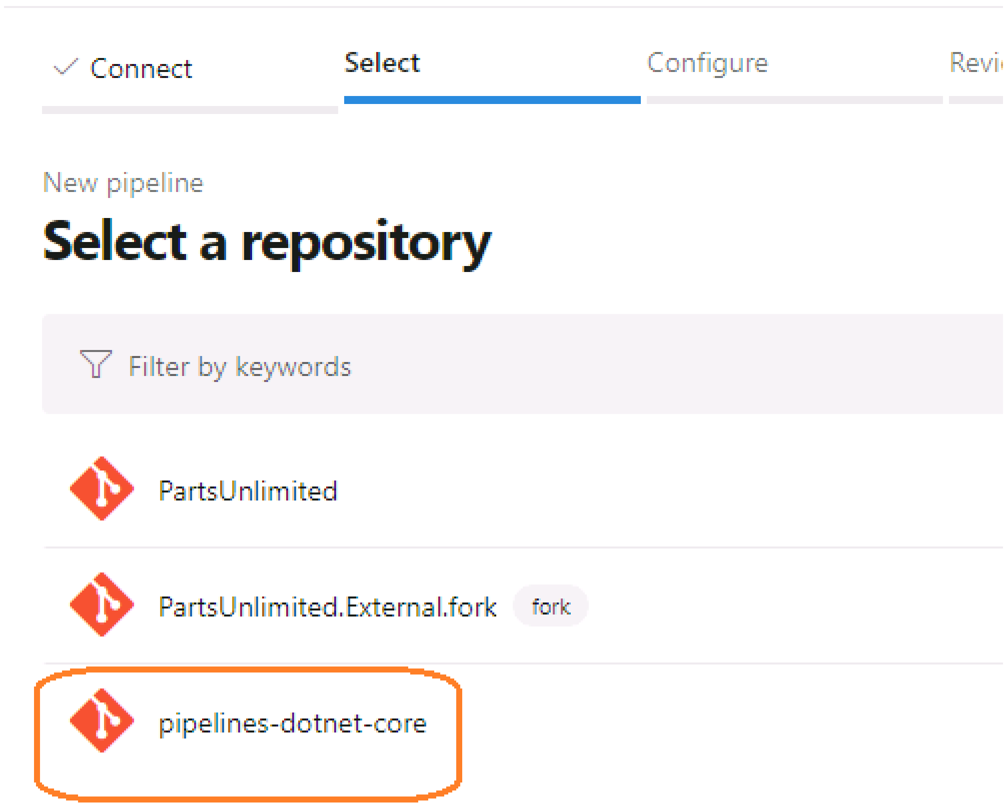
Other Git
Any generic Git repository

Subversion
Centralized version control by Apache

Team Foundation Version Control
Centralized version control repositories

> Notice that other platforms like Bitbucket, GitHub, etc. are also supported.

4. Select the **pipelines-dotnet-core** repository.



New pipeline

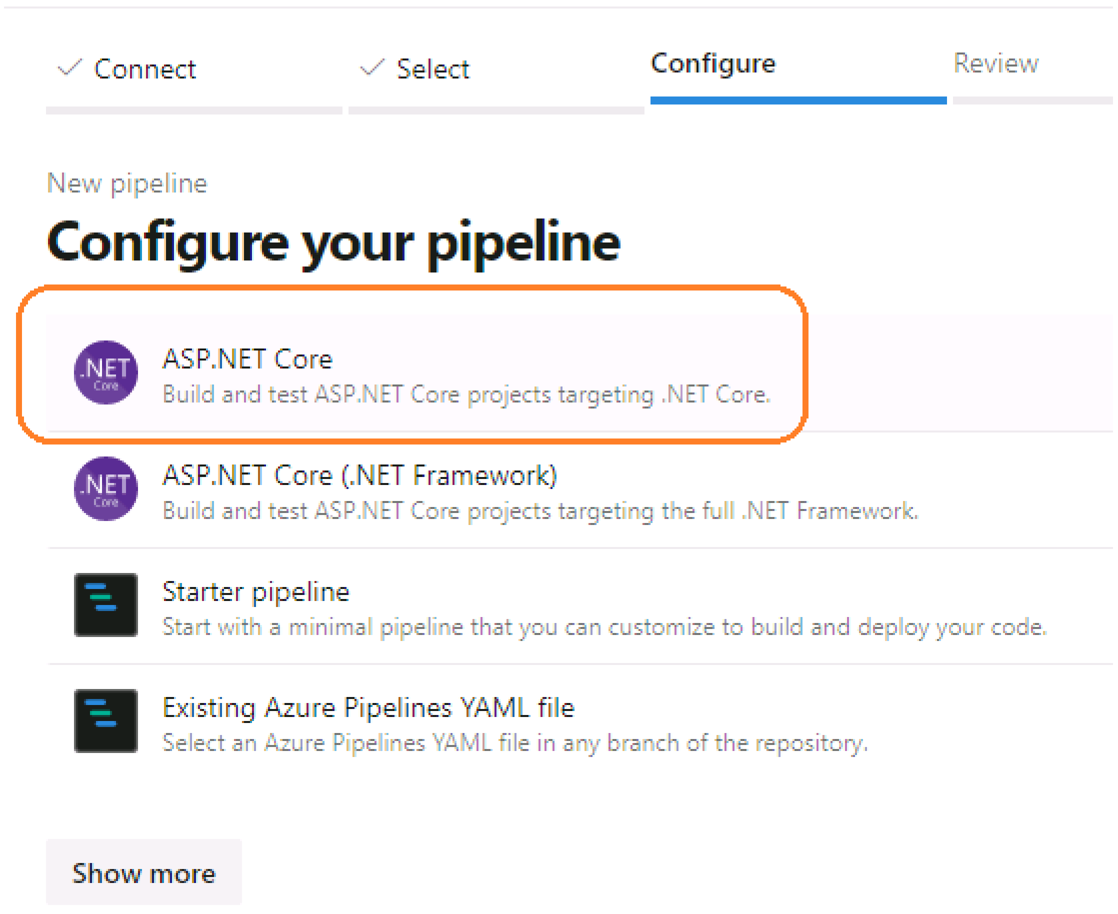
Select a repository

PartsUnlimited

PartsUnlimited.External.fork fork

pipelines-dotnet-core

5. Select the **ASP.NET Core** template as the starting point for your pipeline.



New pipeline

Configure your pipeline

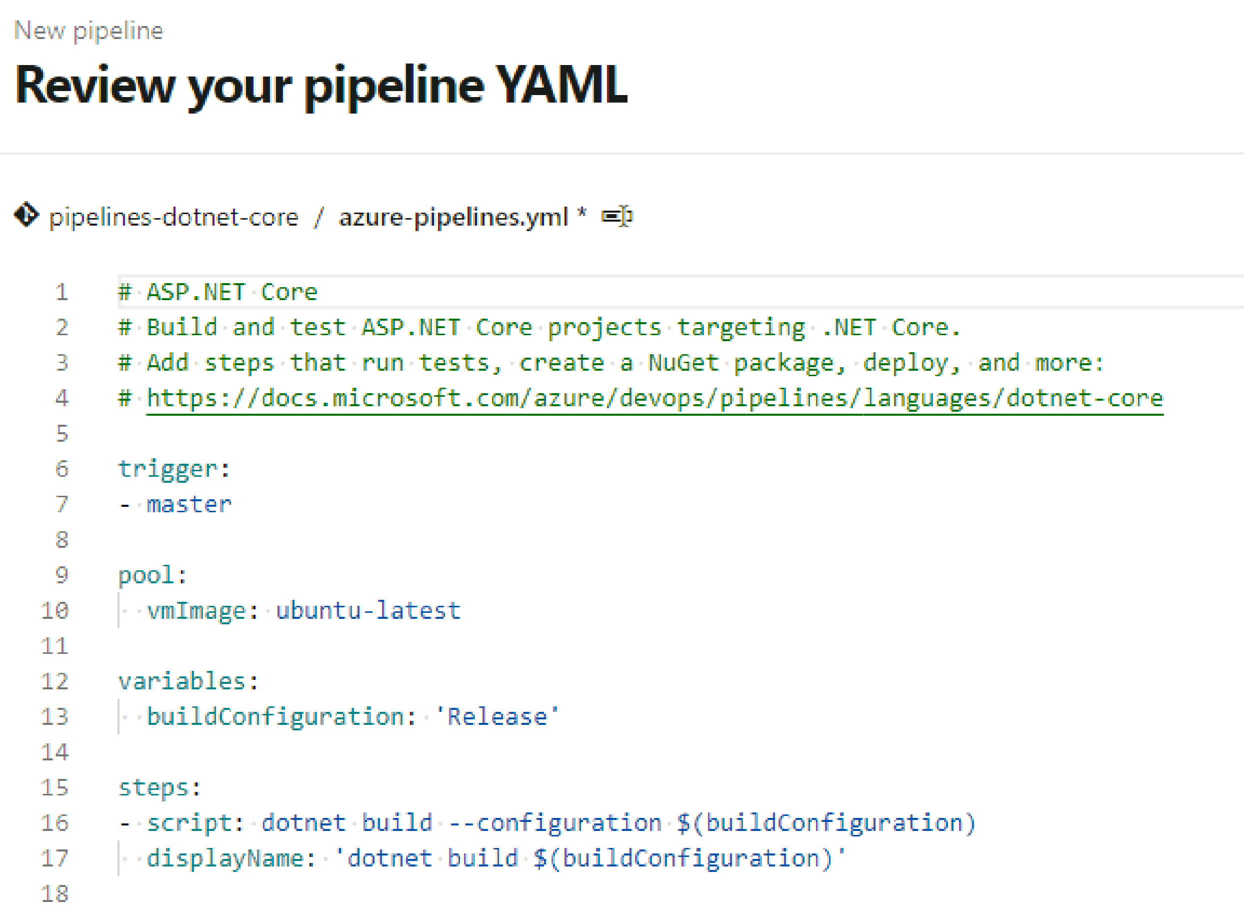
ASP.NET Core
Build and test ASP.NET Core projects targeting .NET Core.

ASP.NET Core (.NET Framework)
Build and test ASP.NET Core projects targeting the full .NET Framework.

Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.

Existing Azure Pipelines YAML file
Select an Azure Pipelines YAML file in any branch of the repository.

6. Review the contents of the YAML definition. It will be saved as a new file called “azure-pipelines.yml” in the root of the repository and contain everything needed to build and deploy a typical ASP.NET Core solution. You can also customize the YAML as needed.


New pipeline

Review your pipeline YAML

pipelines-dotnet-core / azure-pipelines.yml *

```yaml
# ASP.NET Core
# Build and test ASP.NET Core projects targeting .NET Core.
# Add steps that run tests, create a NuGet package, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core

trigger:
- master

pool:
  vmImage: ubuntu-latest

variables:
  buildConfiguration: 'Release'

steps:
- script: dotnet build --configuration $(buildConfiguration)
  displayName: 'dotnet build $(buildConfiguration)'

```

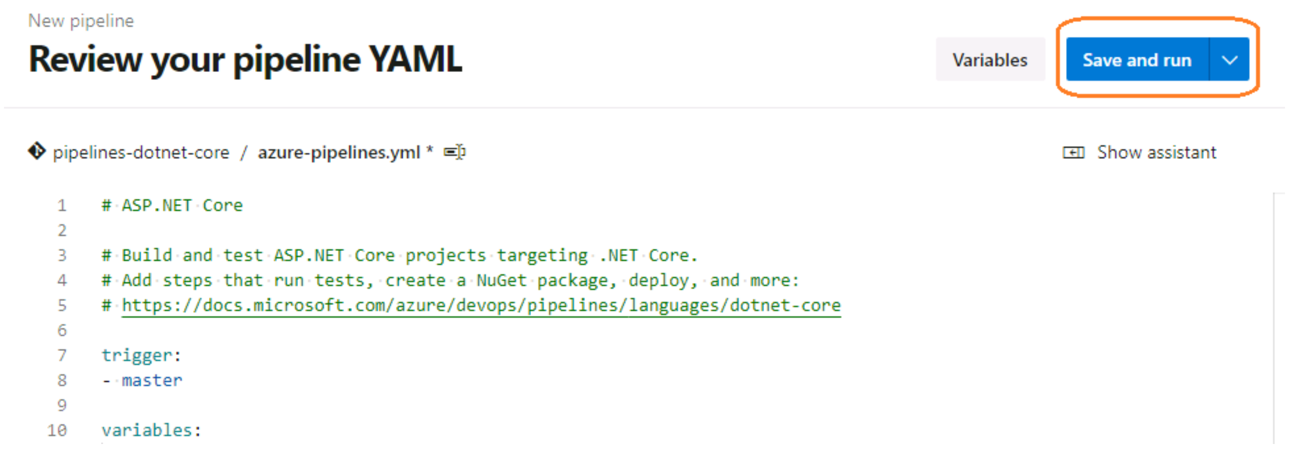
7. Remove all the contents of the YAML definitions and **copy and paste** the code below into your YAML definition.

> Keep the indentation intact. Note that there is a **Build stage** defined in the YAML pipeline. You can define whatever stages you need to better organize and track pipeline progress. You can also use the **task assistant** to add the tasks below. (.NET Core task for Build, Test, and Publish & Publish build artifacts)

```


# ASP.NET Core

# Build and test ASP.NET Core projects targeting .NET Core.
# Add steps that run tests, create a NuGet package, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core

trigger:
- master

variables:
    buildConfiguration: 'Release'

stages:
- stage: Build
  jobs:
  - job: Build
```

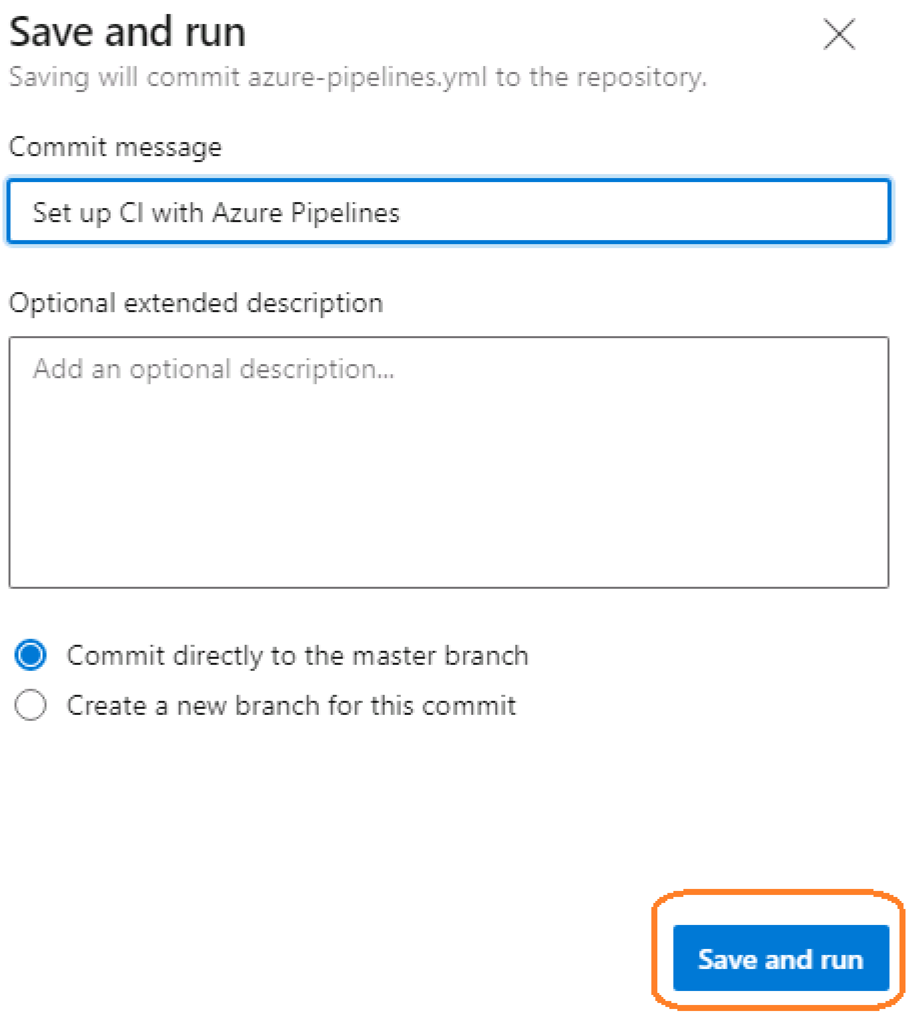
```
    pool:
      name: default
    steps:
    - script: dotnet restore
    - task: DotNetCoreCLI@2
      displayName: Build
      inputs:
        command: build
        projects: '**/*.csproj'
        arguments: '--configuration Release'
    - task: DotNetCoreCLI@2
      displayName: Test
      inputs:
        command: test
        projects: '**/*Tests/*.csproj'
        arguments: '--configuration $(BuildConfiguration)'
    - task: DotNetCoreCLI@2
      displayName: Publish
      inputs:
        command: publish
        publishWebProjects: True
        arguments: '--configuration $(BuildConfiguration) --output
$(build.artifactstagingdirectory)'
        zipAfterPublish: True
    - task: PublishBuildArtifacts@1
      displayName: 'Publish Artifact'
      inputs:
        PathtoPublish: '$(build.artifactstagingdirectory)'
        ArtifactName: 'drop'
    condition: succeededOrFailed()
```

8. Click **Save and run**.


New pipeline
Review your pipeline YAML
Variables
Save and run
pipelines-dotnet-core / azure-pipelines.yml *
Show assistant

```
# ASP.NET Core

# Build and test ASP.NET Core projects targeting .NET Core.
# Add steps that run tests, create a NuGet package, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core

trigger:
- master

variables:
```

9. Click **Save and run** to confirm the commit.


Save and run

Saving will commit azure-pipelines.yml to the repository.

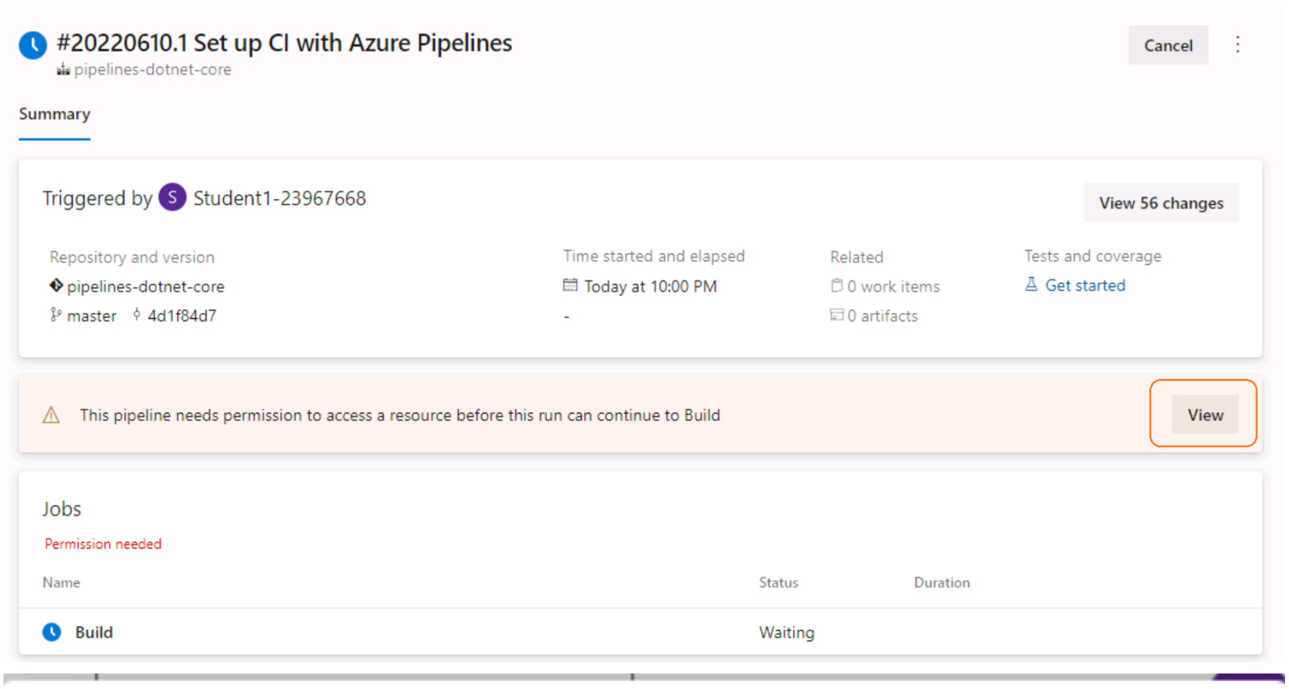
Commit message: Set up CI with Azure Pipelines

Optional extended description: Add an optional description...

[✓ Commit directly to the master branch]

Create a new branch for this commit

10. A message will be displayed asking permission to access the Default build agent pool (if you do not see it, **click** on the running **Build job** to navigate to the logs page to view the message). Click on **View** and click on **Permit**.



#20220610.1 Set up CI with Azure Pipelines

pipelines-dotnet-core

Summary

Triggered by Student1-23967668

View 56 changes

Repository and version: pipelines-dotnet-core, master, 4d1f84d7

Time started and elapsed: Today at 10:00 PM, -

Related: 0 work items, 0 artifacts

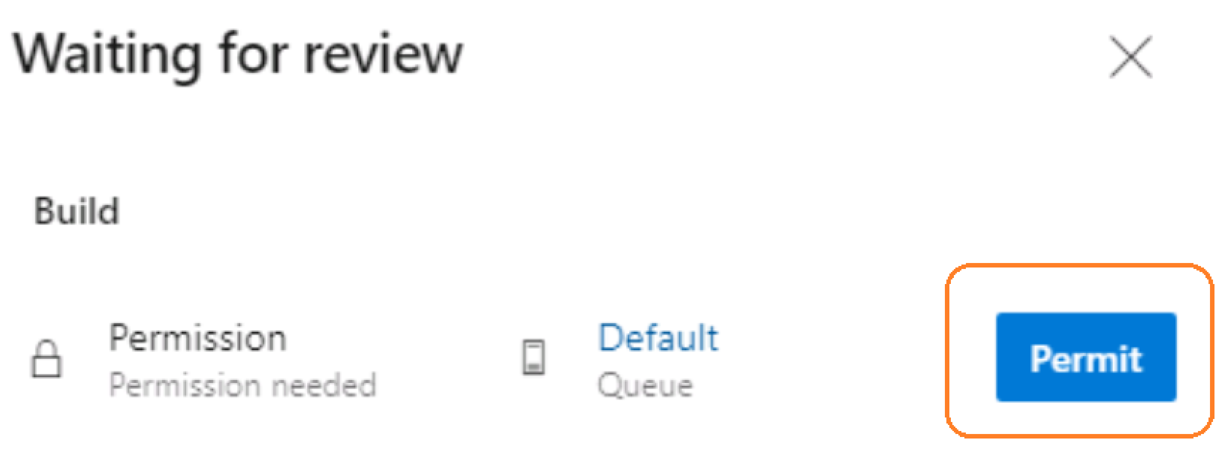
Tests and coverage: Get started

This pipeline needs permission to access a resource before this run can continue to Build

Jobs

Permission needed

Name	Status	Duration
Build	Waiting	



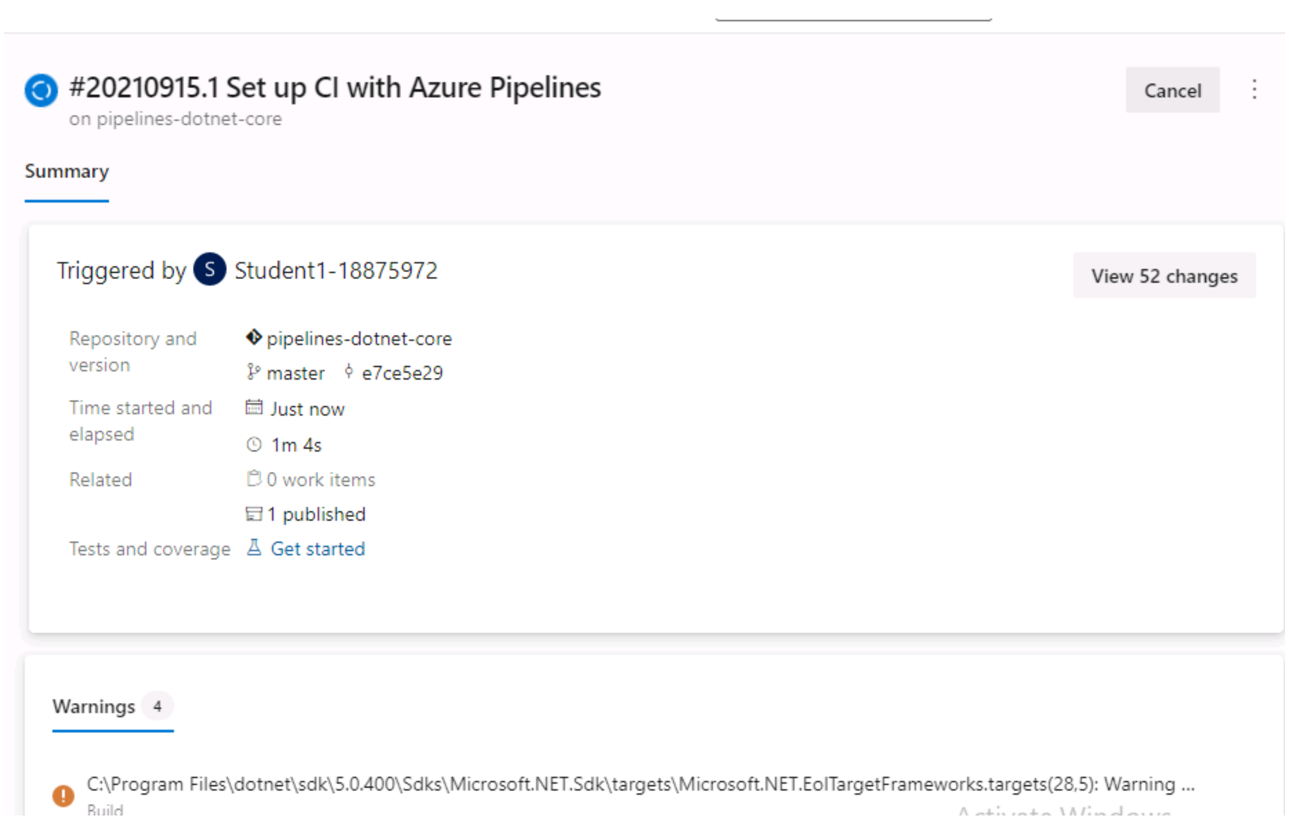
Waiting for review

Build

Permission — Permission needed

Default — Queue

11. Track the build until it completes. Click **Jobs | Build** to see the logs.

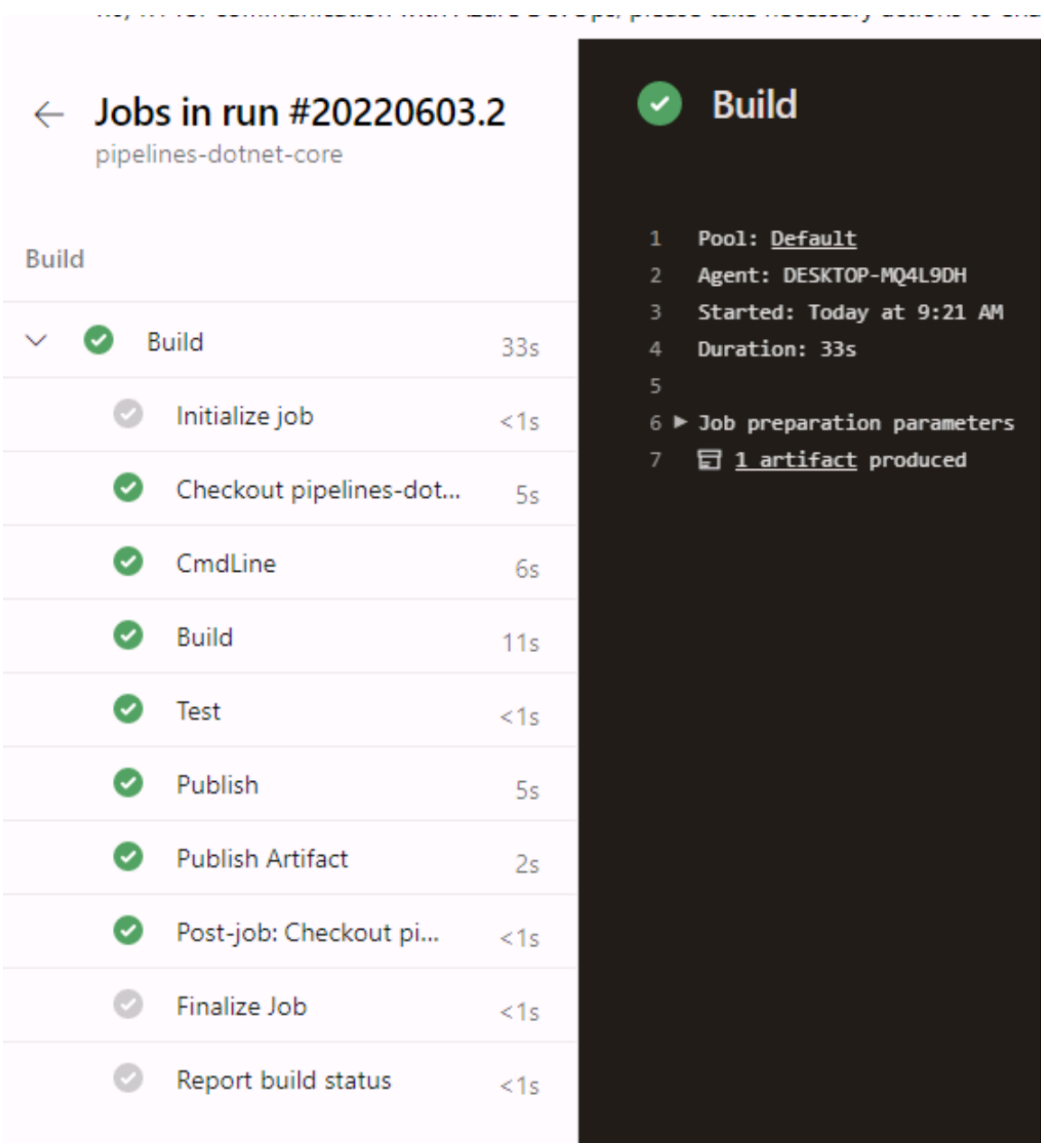


Triggered by Student1-18875972

Repository and version	pipelines-dotnet-core master e7ce5e29
Time started and elapsed	Just now 1m 4s
Related	0 work items 1 published
Tests and coverage	Get started

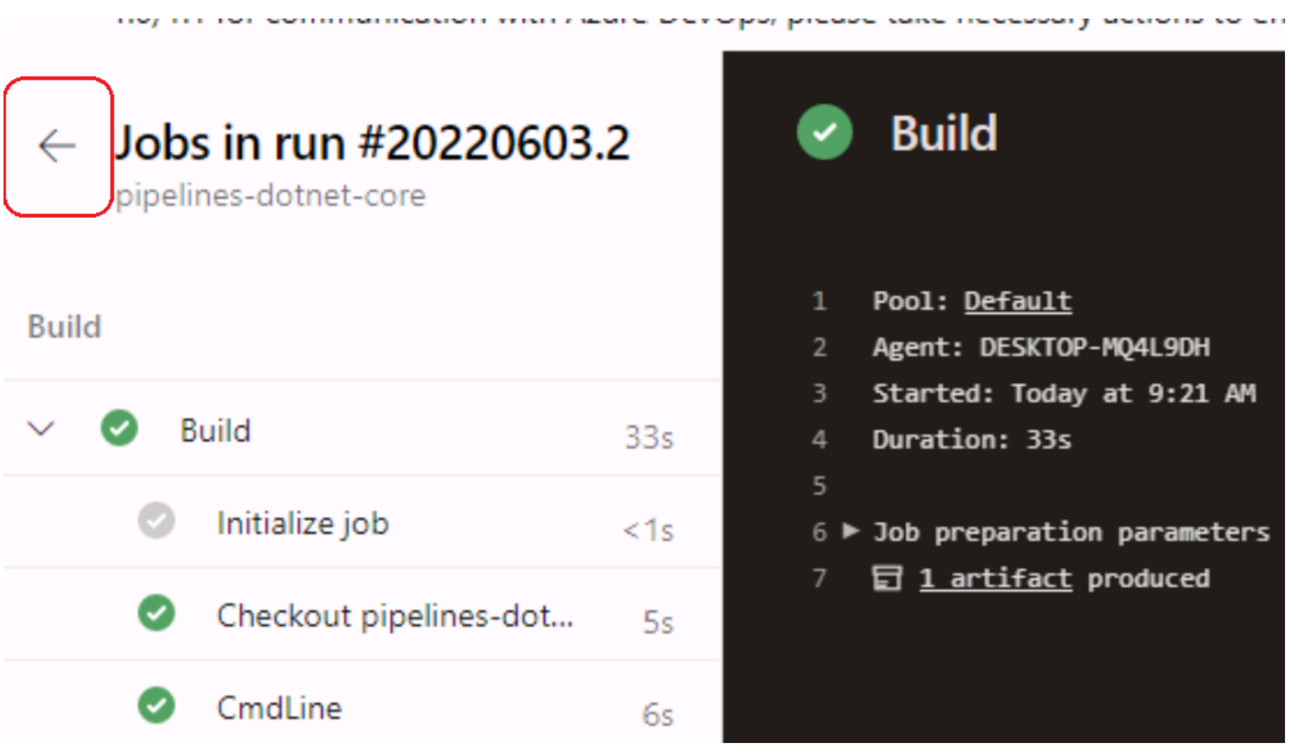
```
C:\Program Files\dotnet\sdk\5.0.400\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.EolTargetFrameworks.targets(28,5): Warning ...
Build
```

12. Each task from the YAML file is available for review, including any warnings and errors.

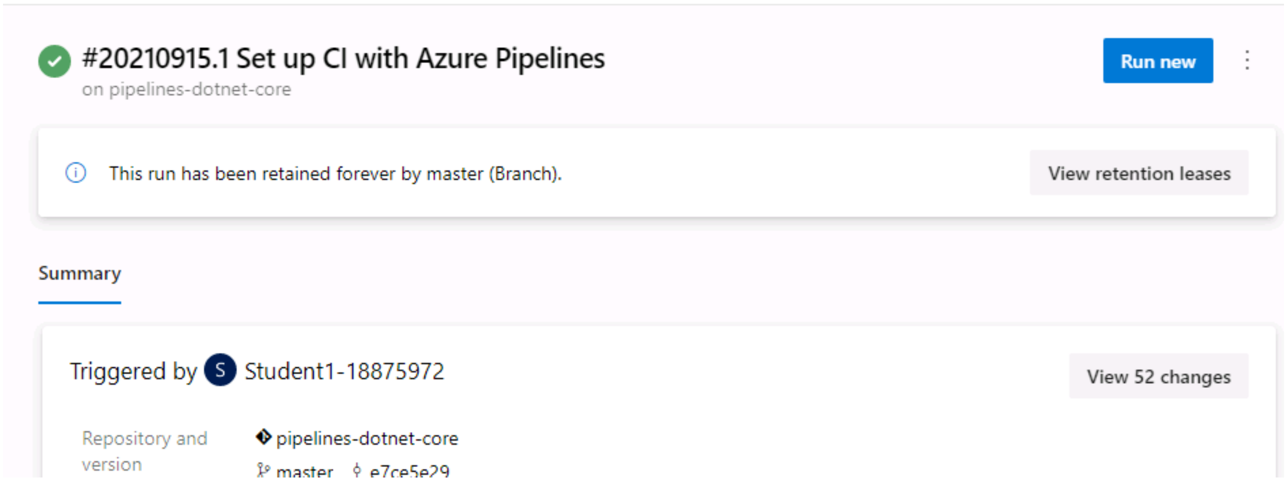


```
Pool: Default
Agent: DESKTOP-MQ4L9DH
Started: Today at 9:21 AM
Duration: 33s

▶ Job preparation parameters
1 artifact produced
```

13. Click the **Back button** to close the tasks view.



```
Pool: Default
Agent: DESKTOP-MQ4L9DH
Started: Today at 9:21 AM
Duration: 33s

▶ Job preparation parameters
1 artifact produced
```



This run has been retained forever by master (Branch).

Triggered by S Student1-18875972

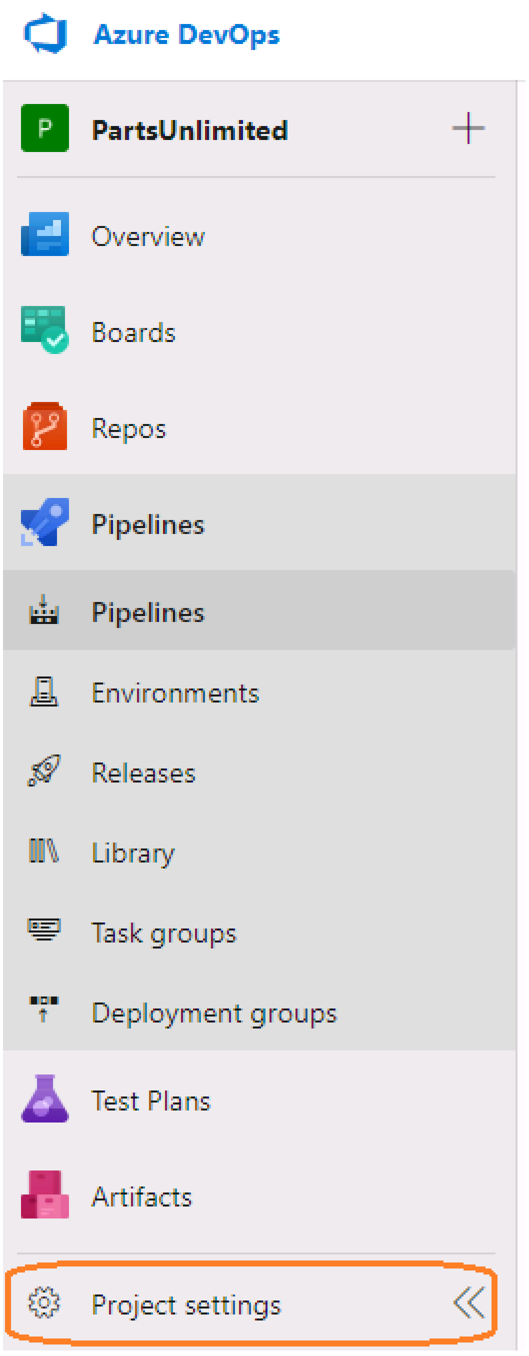
Repository and version: pipelines-dotnet-core, master, e7ce5e29

Module 4: **Azure Pipelines**, **Lab 1: Configuring CI/CD Pipelines**, Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

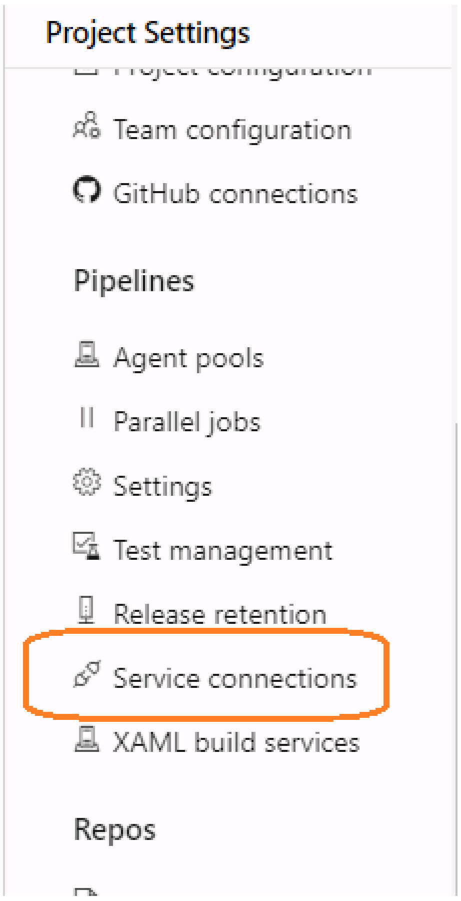
---

Task 5: Setting up a Service Connection to Azure

1. For deploying to the web app we created in the first task, we first need to set up a Service Connection. Select **Project settings** from bottom-left corner of PartsUnlimited project.

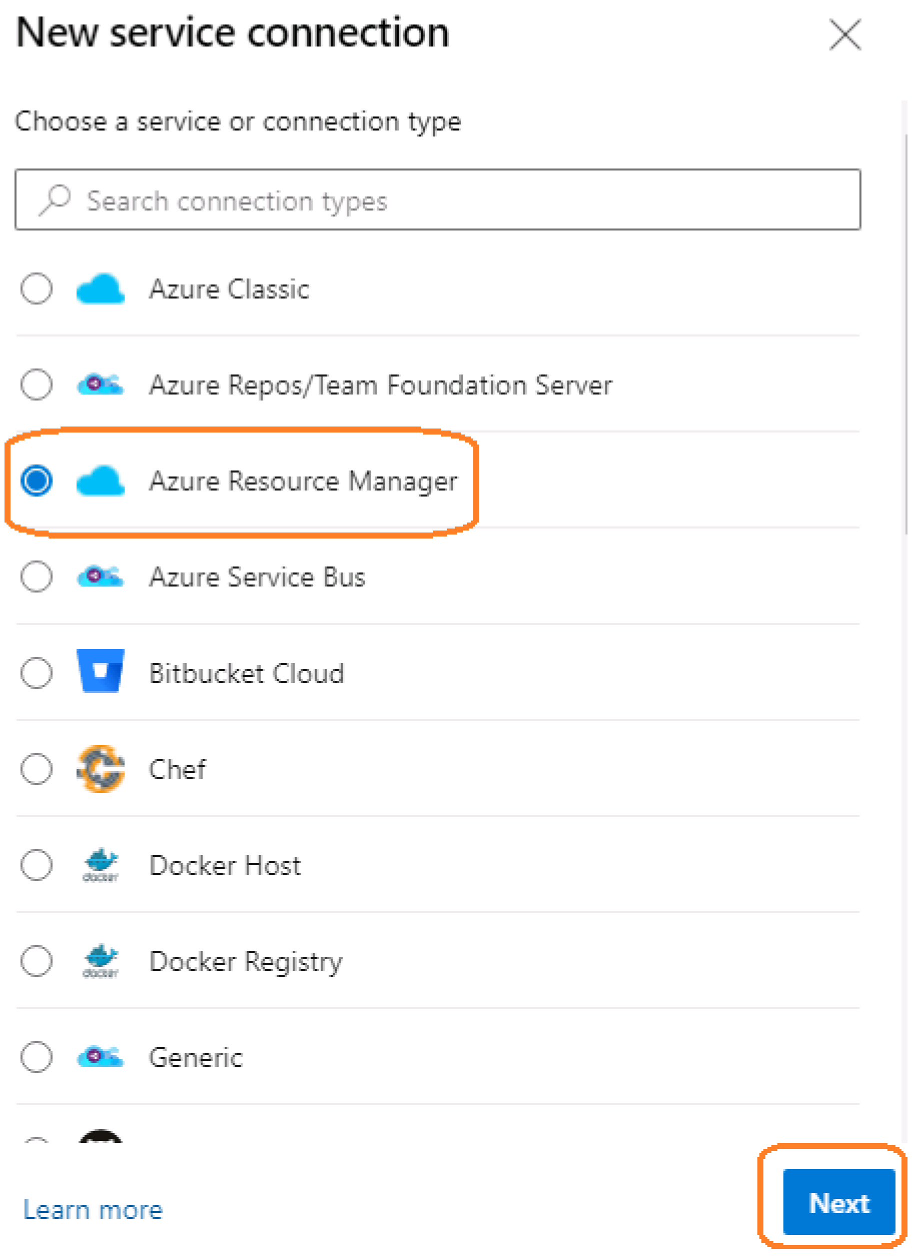


2. Scroll down and click on **Service connection**.



3. Click on **Create service connection**.

4. In the **New service connection** select **Azure Resource Manager** and click **Next**.


New service connection

Choose a service or connection type

Search connection types

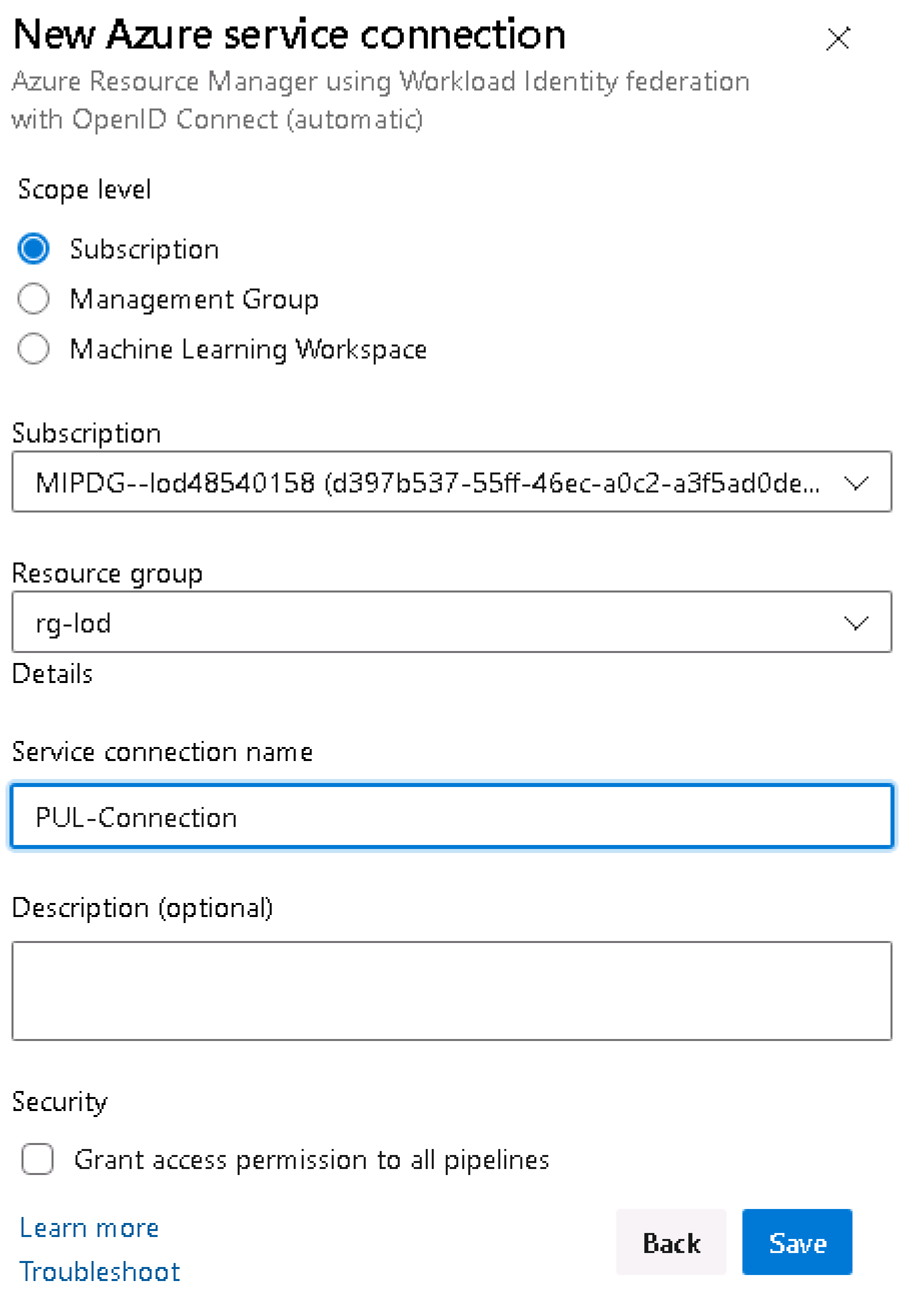
- ( ) Azure Classic
- ( ) Azure Repos/Team Foundation Server
- [✓ Azure Resource Manager]
- ( ) Azure Service Bus
- ( ) Bitbucket Cloud
- ( ) Chef
- ( ) Docker Host
- ( ) Docker Registry
- ( ) Generic

5. Select the recommended default option of **Workload Identity federation (automatic)** and click **Next**.

6. In the **New Azure service connection** pane, enter following information:

   - Scope level: **Subscription**
   - Subscription: **Your Azure Subscription**
   - Resource group: **rg-lod**
   - Service connection name: **PUL-Connection**
   - Click **Save**

> If you don't see Azure Subscription, sign out from Azure DevOps Services and sign back in. Alternatively, try clearing the browser cookies or switching to a difference browser. If this doesn't work then try disconnecting the organization from Microsoft Entra and reconnecting it from the **Organization settings**.



New Azure service connection

Azure Resource Manager using Workload Identity federation with OpenID Connect (automatic)

Scope level

- [✓ Subscription]
- Management Group
- Machine Learning Workspace

Subscription: MIPDG--lod48540158 (d397b537-55ff-46ec-a0c2-a3f5ad0de...

Resource group: rg-lod

Details

Service connection name: PUL-Connection

Description (optional):

Security

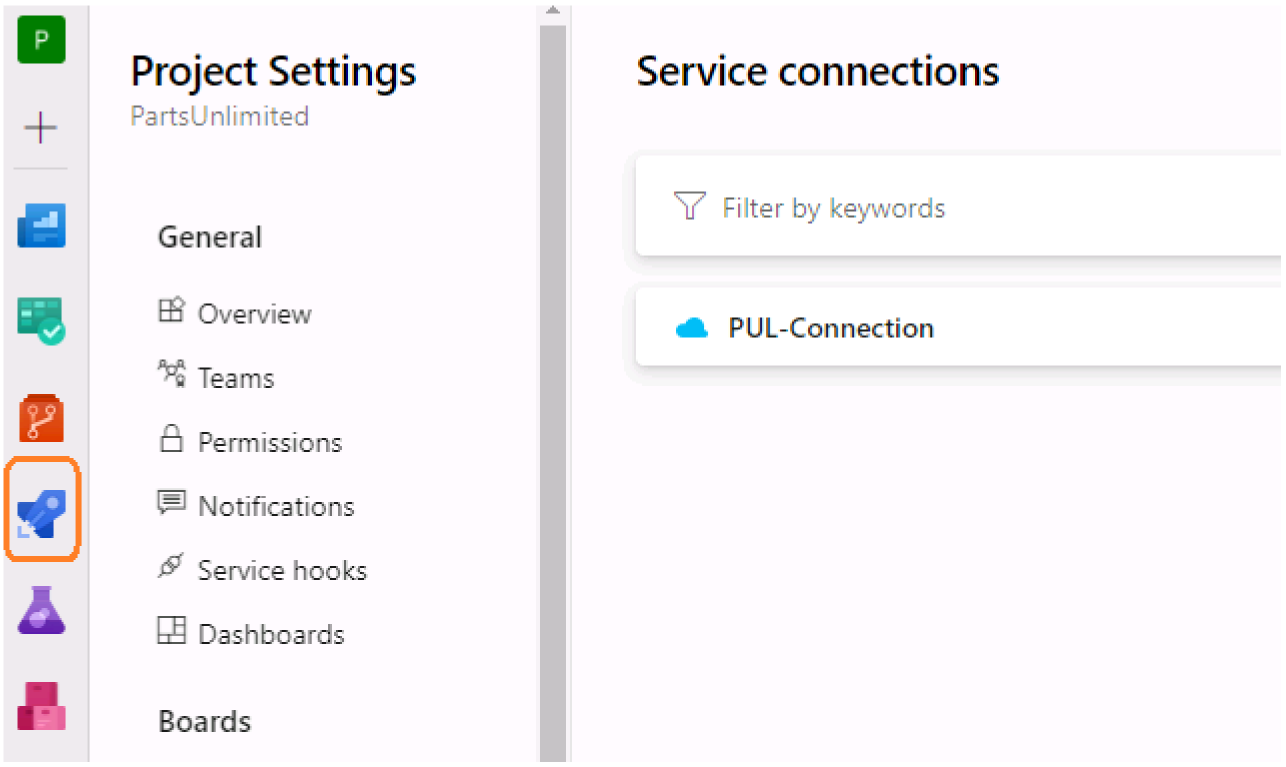
- [ ] Grant access permission to all pipelines

Module 4: **Azure Pipelines**, **Lab 1: Configuring CI/CD Pipelines**, Exercise 1: Configuring CI/CD Pipelines as Code With YAML in Azure DevOps

Task 6: Adding continuous delivery to the YAML definition

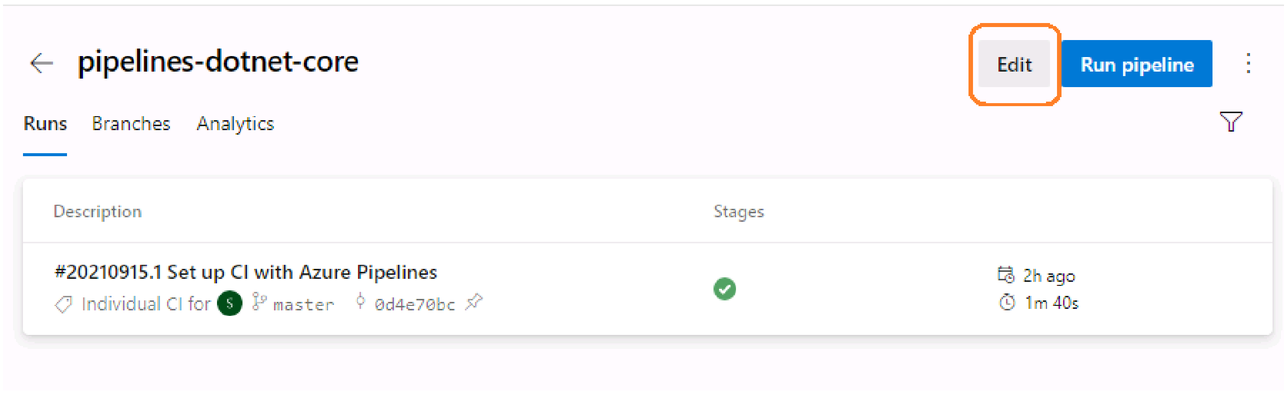
1. Click on **Pipelines** to switch back to the Azure Pipelines.

> Now that the Service Connection to Azure resource group is set up and the build and test definitions are in place, we can now add YAML definitions for deployment of the application to Azure.



2. Click on **pipelines-dotnet-core** pipeline that you started working on in the earlier task.

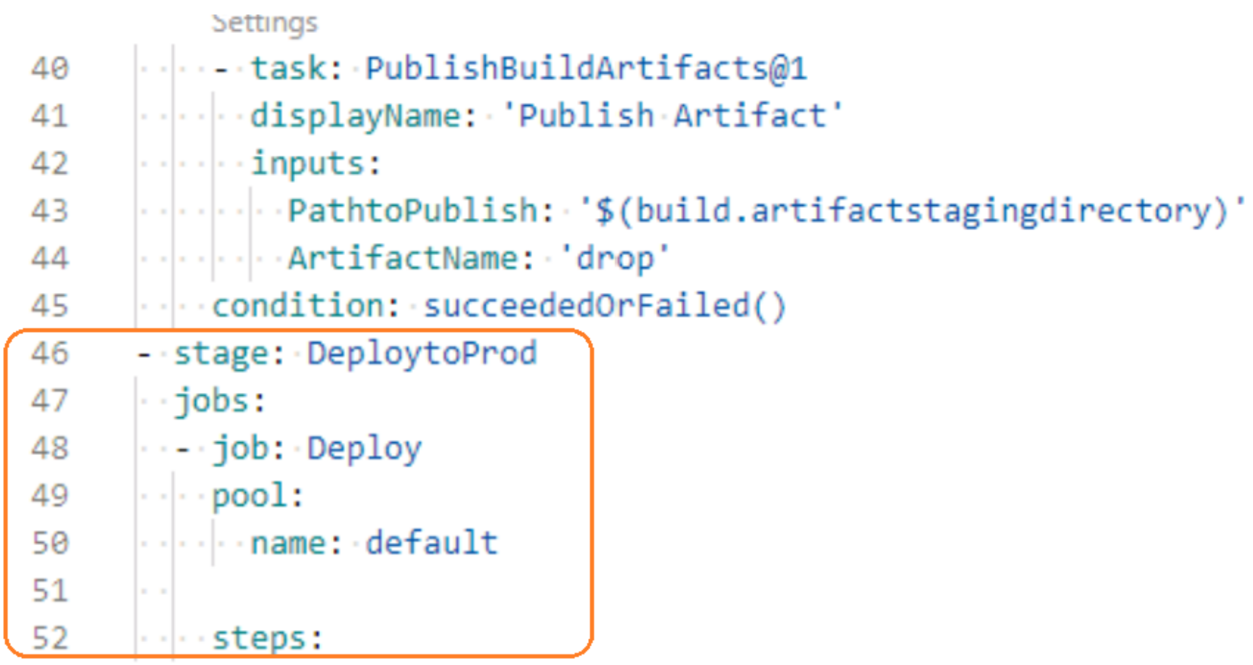
3. Click on **Edit** to start editing the YAML definitions.



4. At the bottom of the file, on the new line add the configuration below to define a second stage.

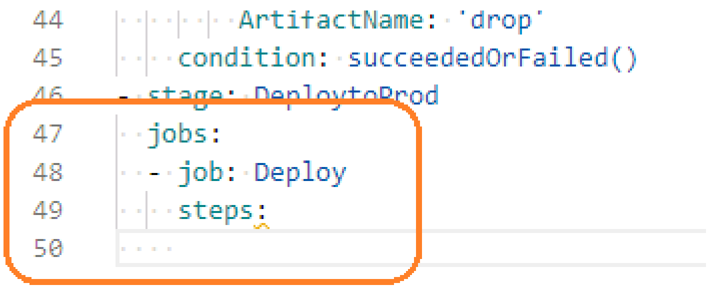
> Make sure you start at the beginning of the new line and keep the indentation intact.

```
- stage: DeploytoProd
  jobs:
  - job: Deploy
    pool:
        name: default
    steps:
```



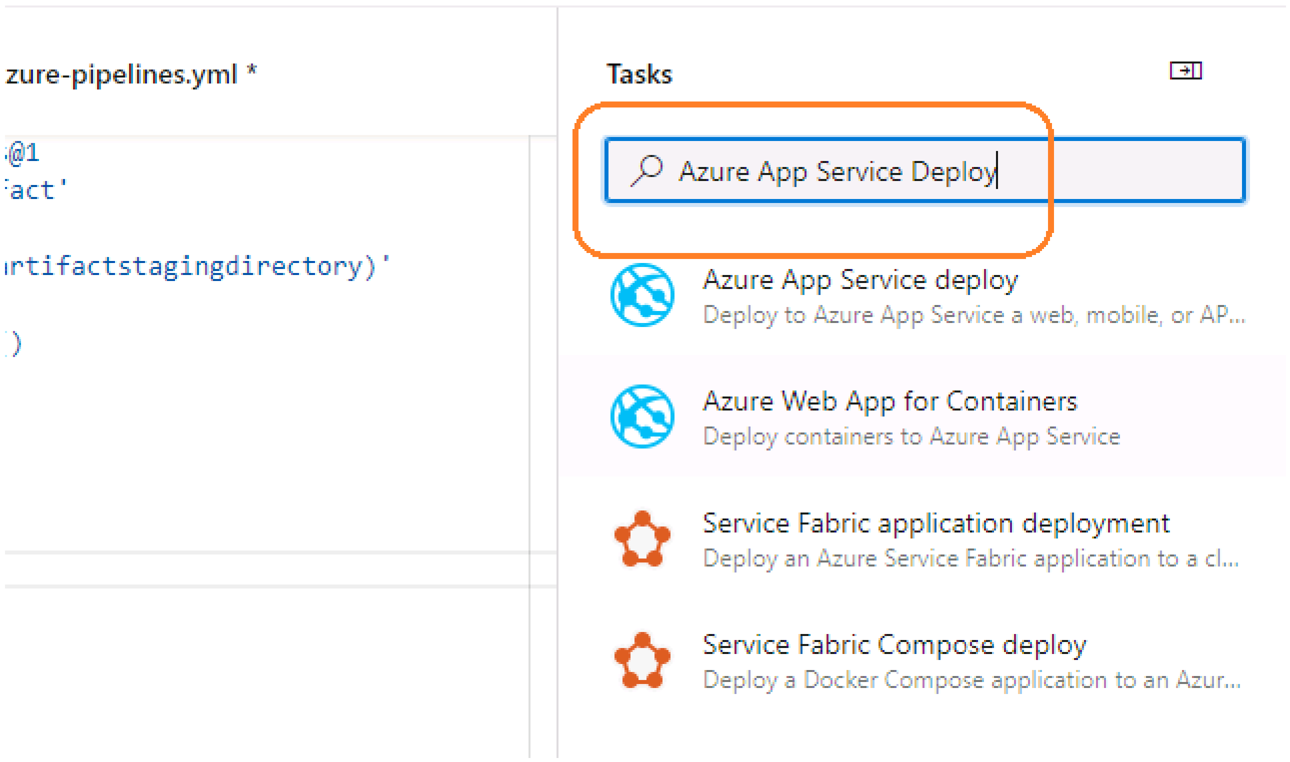
```
    - task: PublishBuildArtifacts@1
      displayName: 'Publish Artifact'
      inputs:
        PathtoPublish: '$(build.artifactstagingdirectory)'
        ArtifactName: 'drop'
    condition: succeededOrFailed()
- stage: DeploytoProd
  jobs:
  - job: Deploy
    pool:
      name: default

    steps:
```

5. Set the cursor on a new line at the end of the YAML definition. This will be the location where new tasks will be added.



```
        ArtifactName: 'drop'
    condition: succeededOrFailed()
- stage: DeploytoProd
  jobs:
  - job: Deploy
    steps:

```

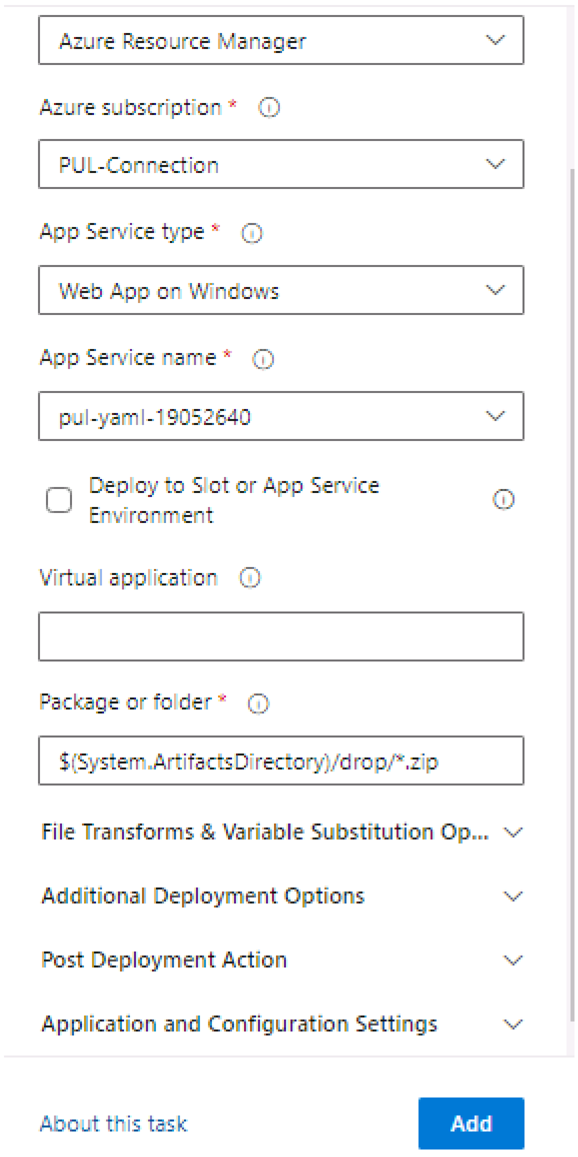
6. Place the curson in **Search tasks** under **Tasks** and start typing **Azure App Service Deploy** task.


zure-pipelines.yml *
Tasks
Azure App Service Deploy
Azure App Service deploy
Deploy to Azure App Service a web, mobile, or AP...
Azure Web App for Containers
Deploy containers to Azure App Service
Service Fabric application deployment
Deploy an Azure Service Fabric application to a cl...
Service Fabric Compose deploy
Deploy a Docker Compose application to an Azur...

```
@1
act'

rtifactstagingdirectory)'

)
```

7. Select **Azure App Service deploy** task and enter following details:

   - Connection type: **Azure Resource Manager**
   - Azure Subscription: **PUL-Connection**
   - Click **Authorize** if needed and follow the path to complete authorization.
   - App Service name: **pul-yaml-[YourInitials]**
   - Package or folder: **$(System.ArtifactsDirectory)/drop/*.zip**
   - Click **Add**


Azure Resource Manager

Azure subscription *

PUL-Connection

App Service type *

Web App on Windows

App Service name *

pul-yaml-19052640

Deploy to Slot or App Service Environment

Virtual application

Package or folder *

$(System.ArtifactsDirectory)/drop/*.zip

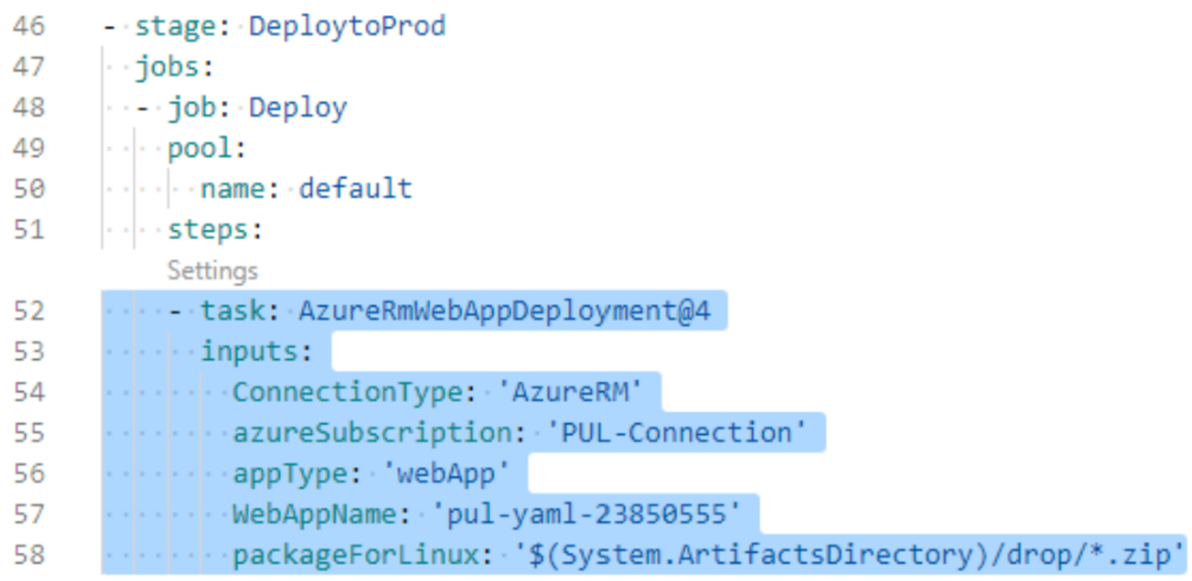
File Transforms & Variable Substitution Op...

Additional Deployment Options

Post Deployment Action

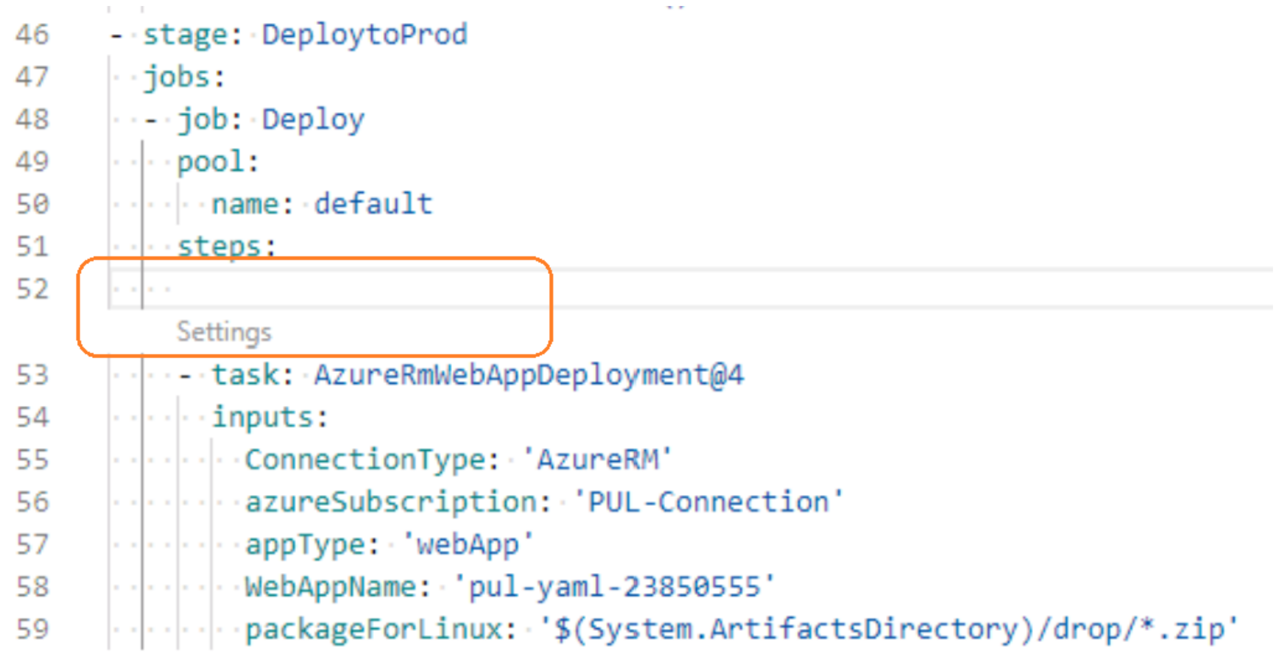
Application and Configuration Settings

8. This will add YAML definition for the App Service Deploy task to the cursor location in the file. You may or may not have to indent based on your cursor location.


```yaml
- stage: DeploytoProd
  jobs:
  - job: Deploy
    pool:
      name: default
    steps:
    Settings
    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'PUL-Connection'
        appType: 'webApp'
        WebAppName: 'pul-yaml-23850555'
        packageForLinux: '$(System.ArtifactsDirectory)/drop/*.zip'
```

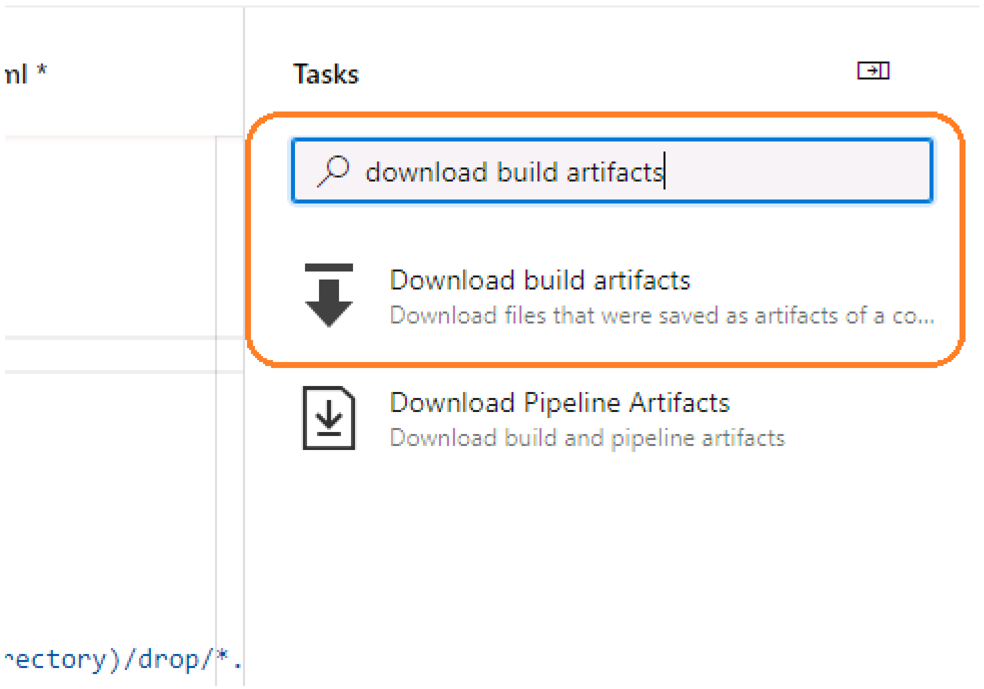
9. Place the cursor on the first line under the **steps node** of the deployment stage. This is where we will add a **Download build artifacts** task.

   > It's important to note that the two stages (Build and Deploy) in this YAML file will run independently. As a result, the build output from the first (Build) stage will not be available to the second (Deploy) stage without special consideration. It is for this reason, you added the **Publish build artifacts** task at the end of the Build stage and you will add **Download build artifacts** task in this (Deploy stage)

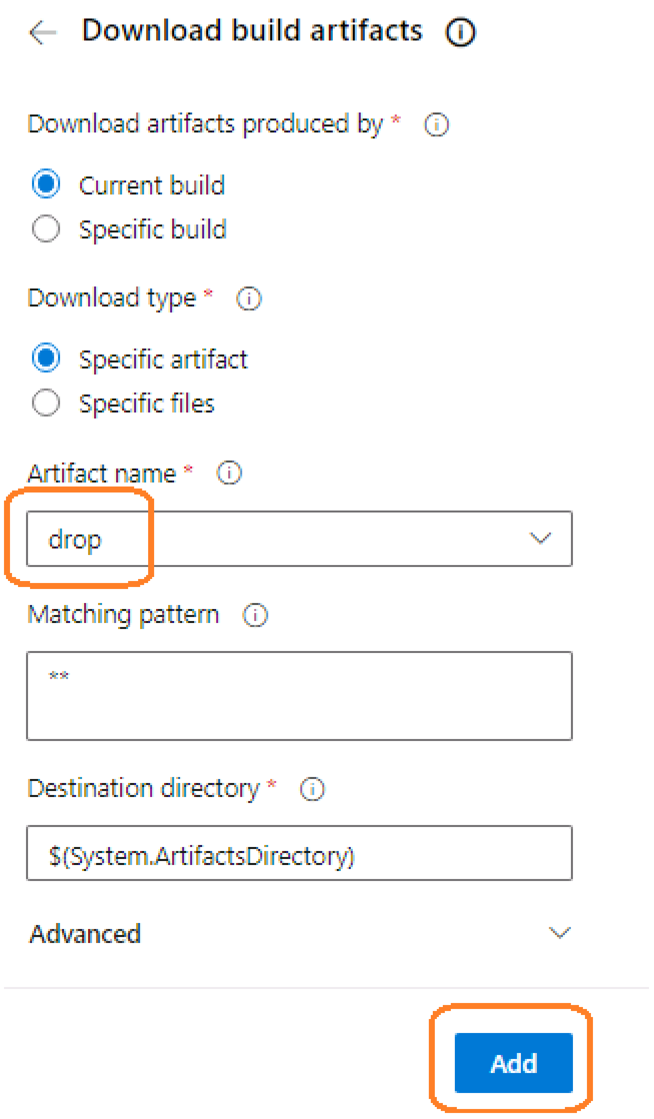


```
- stage: DeploytoProd
  jobs:
  - job: Deploy
    pool:
      name: default
    steps:
    
    Settings
    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'PUL-Connection'
        appType: 'webApp'
        WebAppName: 'pul-yaml-23850555'
        packageForLinux: '$(System.ArtifactsDirectory)/drop/*.zip'
```

10. Under **Tasks** in the Task Assistant, search the task **download build artifacts** and select the **Download build artifacts** task.


nl *
Tasks
download build artifacts
Download build artifacts
Download files that were saved as artifacts of a co...
Download Pipeline Artifacts
Download build and pipeline artifacts
ectory)/drop/*.

11. In the **Download build artifacts** pane, enter the **Artifact name** as **drop**. Use the defaults for everything else and click **Add**.


Download build artifacts

Download artifacts produced by *

[✓ Current build]
Specific build

Download type *

[✓ Specific artifact]
Specific files

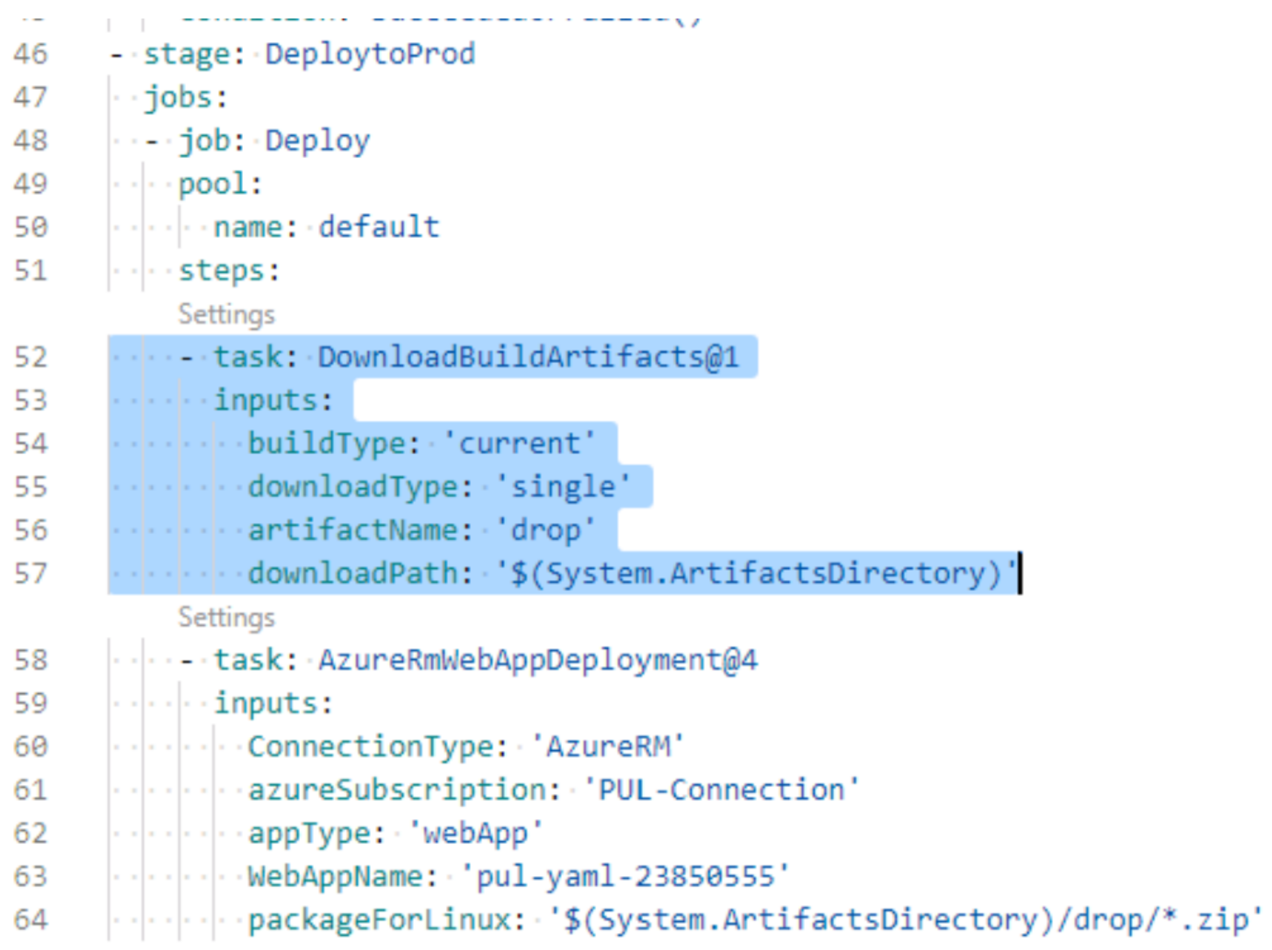
Artifact name *: drop

Matching pattern: **

Destination directory *: $(System.ArtifactsDirectory)

Advanced

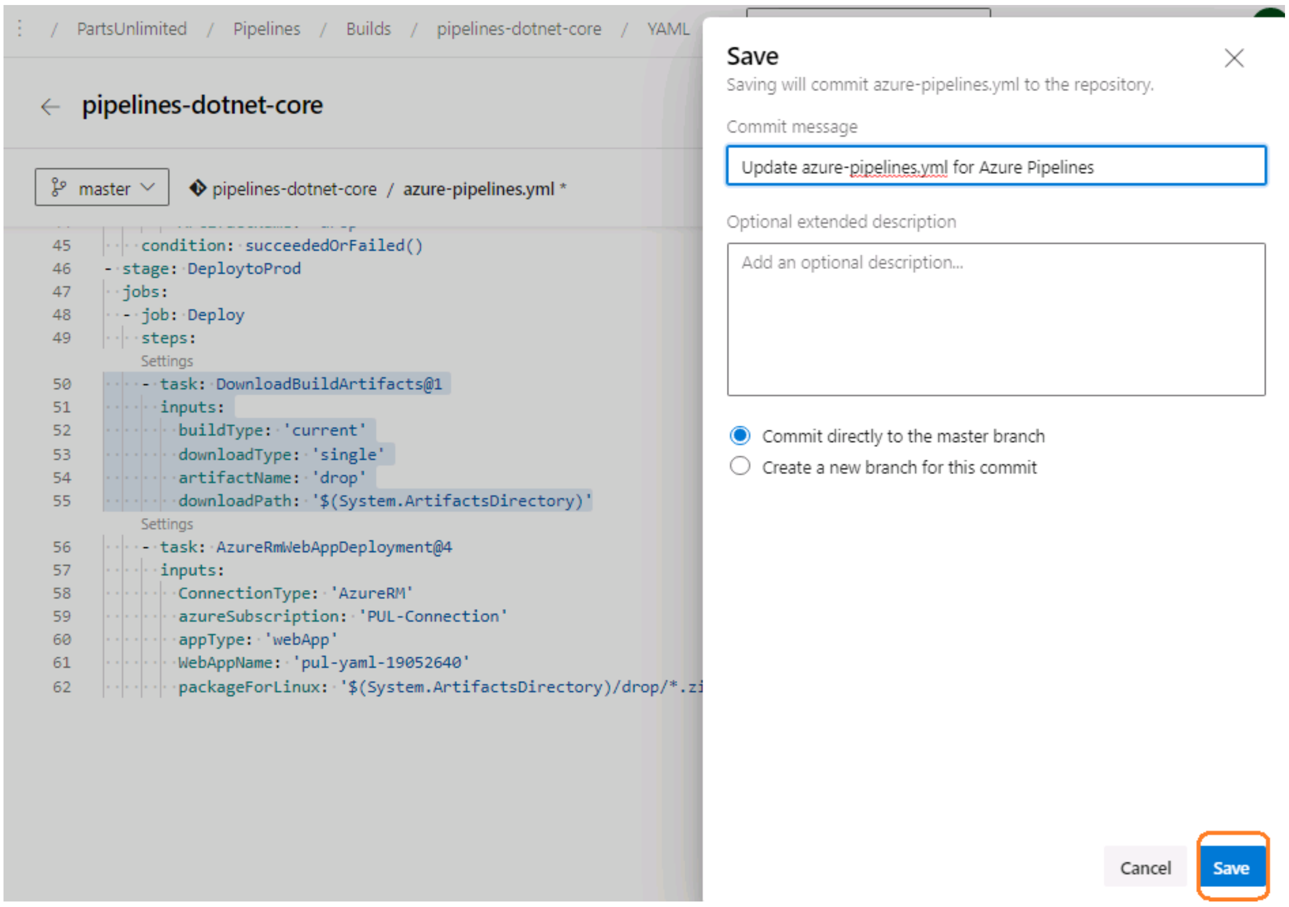
12. Your **deploy** stage should look like below:



```
- stage: DeploytoProd
  jobs:
  - job: Deploy
    pool:
      name: default
    steps:
    Settings
    - task: DownloadBuildArtifacts@1
      inputs:
        buildType: 'current'
        downloadType: 'single'
        artifactName: 'drop'
        downloadPath: '$(System.ArtifactsDirectory)'
    Settings
    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'PUL-Connection'
        appType: 'webApp'
        WebAppName: 'pul-yaml-23850555'
        packageForLinux: '$(System.ArtifactsDirectory)/drop/*.zip'
```

13. Click **Validate and save** from the top-right corner to commit the changes. Since the YAML definition has *trigger* set to *-master*, any change to the master branch that the above changes make will trigger the pipeline to run.

14. Confirm the **Save** when asked. This will begin a new build.



PartsUnlimited / Pipelines / Builds / pipelines-dotnet-core / YAML

pipelines-dotnet-core

master · pipelines-dotnet-core / azure-pipelines.yml *

```
    condition: succeededOrFailed()
- stage: DeploytoProd
  jobs:
  - job: Deploy
    steps:
    Settings
    - task: DownloadBuildArtifacts@1
      inputs:
        buildType: 'current'
        downloadType: 'single'
        artifactName: 'drop'
        downloadPath: '$(System.ArtifactsDirectory)'
    Settings
    - task: AzureRmWebAppDeployment@4
      inputs:
        ConnectionType: 'AzureRM'
        azureSubscription: 'PUL-Connection'
        appType: 'webApp'
        WebAppName: 'pul-yaml-19052640'
        packageForLinux: '$(System.ArtifactsDirectory)/drop/*.zi
```

Save

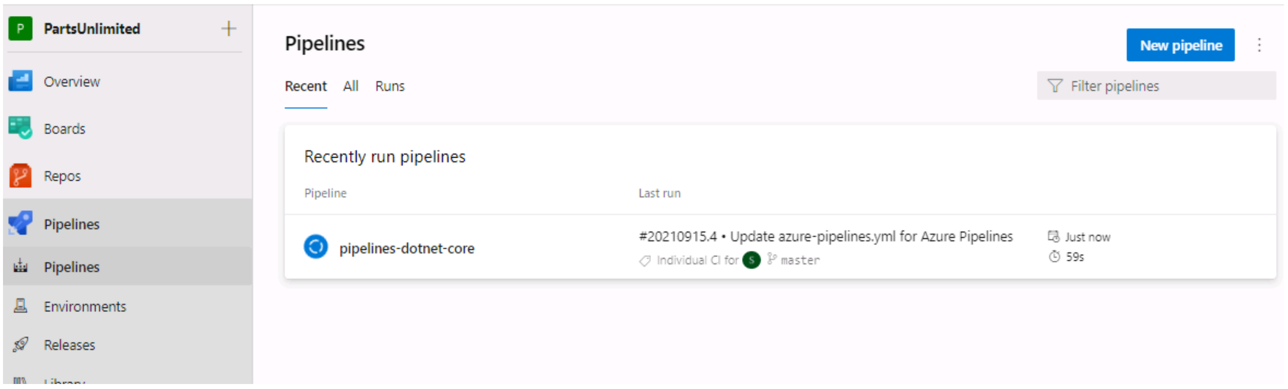
Saving will commit azure-pipelines.yml to the repository.

Commit message: Update azure-pipelines.yml for Azure Pipelines

Optional extended description: Add an optional description...

(●) Commit directly to the master branch
( ) Create a new branch for this commit

15. Return to the Pipelines view by clicking on **Pipelines** from the left pane.



Pipelines

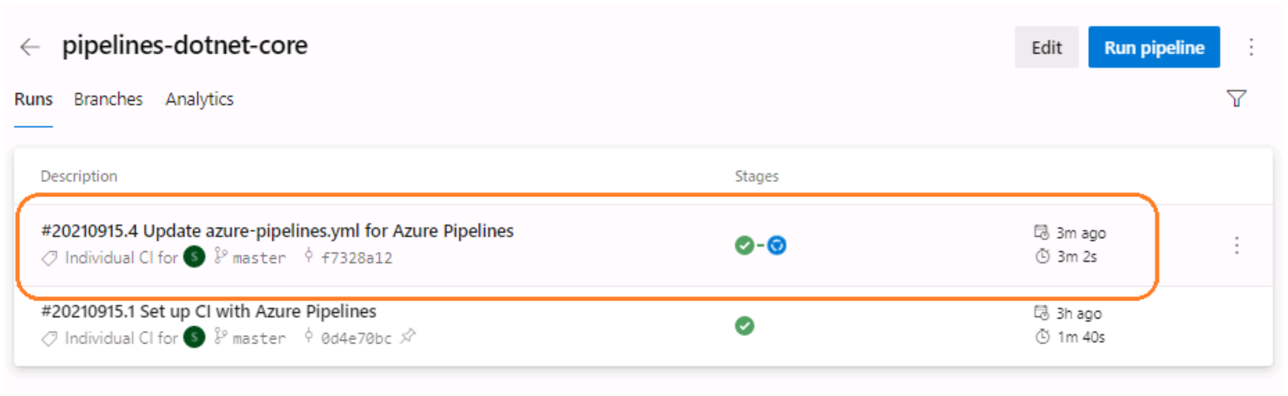
Recently run pipelines

Pipeline	Last run	
pipelines-dotnet-core	#20210915.4 • Update azure-pipelines.yml for Azure Pipelines Individual CI for S master	Just now 59s

16. Click on the pipeline to view the run of the pipeline. Click on the latest run of the pipeline to see the details of the run.

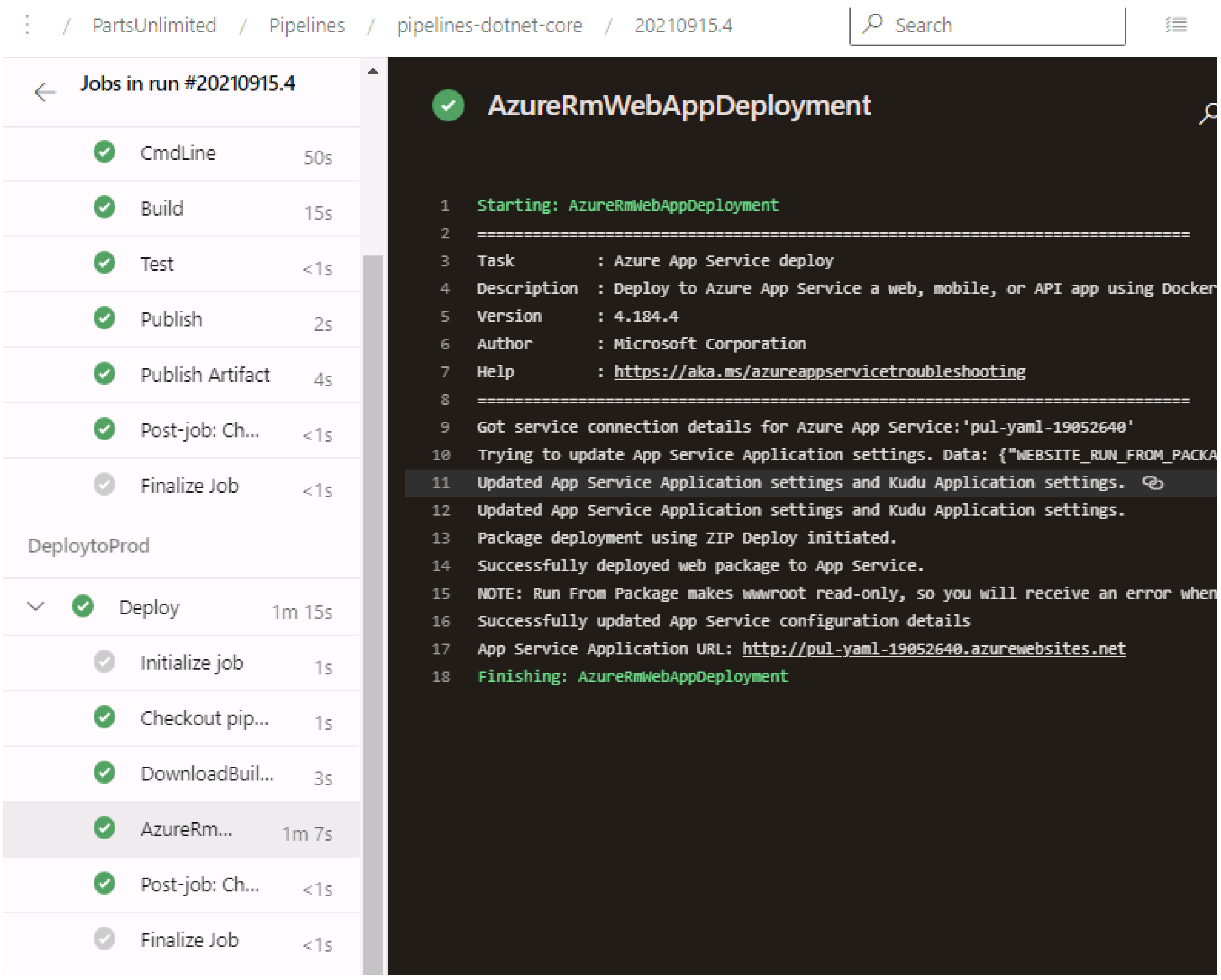
> You may be prompted to authorize the service connection after the Build stage is complete. If prompted, enter the Azure credentials from the first task.

> Notice the difference in the stages displayed for the latest run compared to the previous run. This is because in this run we have two stages.



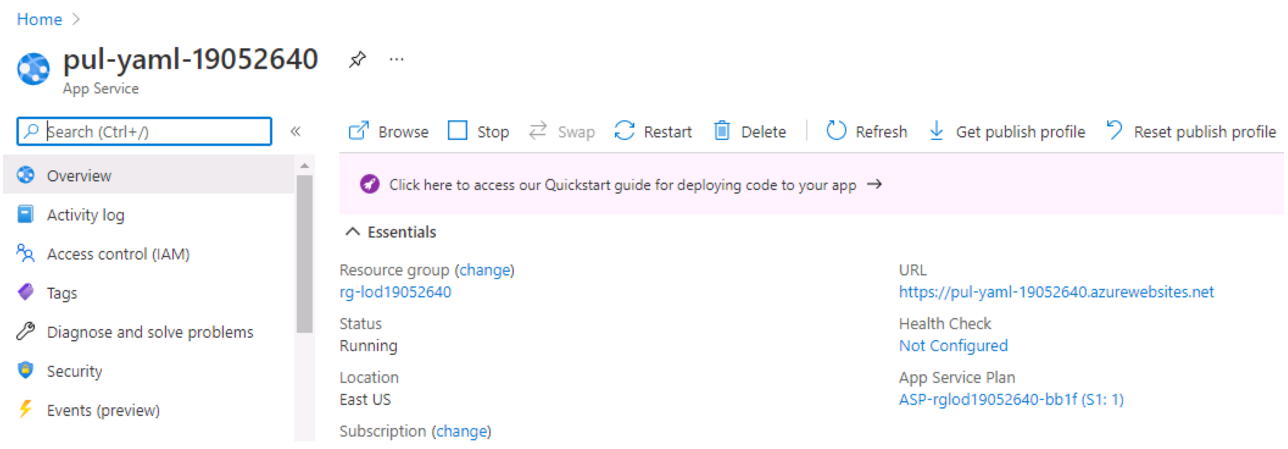
Description	Stages	
#20210915.4 Update azure-pipelines.yml for Azure Pipelines Individual CI for S master f7328a12		3m ago 3m 2s
#20210915.1 Set up CI with Azure Pipelines Individual CI for S master 0d4e70bc		3h ago 1m 40s

17. Scroll down to see **Stages** and **Jobs** . You can see the **Build** and **DeploytoProd** stages. Click on each of those stages to view detailed logs.



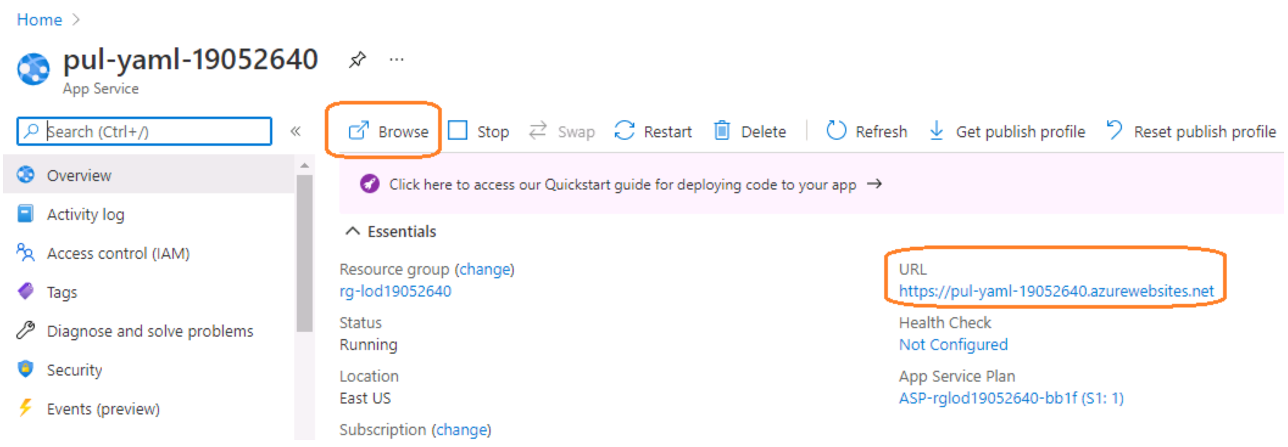
```
Starting: AzureRmWebAppDeployment
==============================================================================
Task         : Azure App Service deploy
Description  : Deploy to Azure App Service a web, mobile, or API app using Docker
Version      : 4.184.4
Author       : Microsoft Corporation
Help         : https://aka.ms/azureappservicetroubleshooting
==============================================================================
Got service connection details for Azure App Service:'pul-yaml-19052640'
Trying to update App Service Application settings. Data: {"WEBSITE_RUN_FROM_PACKA
Updated App Service Application settings and Kudu Application settings.
Updated App Service Application settings and Kudu Application settings.
Package deployment using ZIP Deploy initiated.
Successfully deployed web package to App Service.
NOTE: Run From Package makes wwwroot read-only, so you will receive an error when
Successfully updated App Service configuration details
App Service Application URL: http://pul-yaml-19052640.azurewebsites.net
Finishing: AzureRmWebAppDeployment
```

18. When both the stages run successfully, switch back to the Azure Portal and navigate to the **pul-yaml-[YourInitials]** App Service.

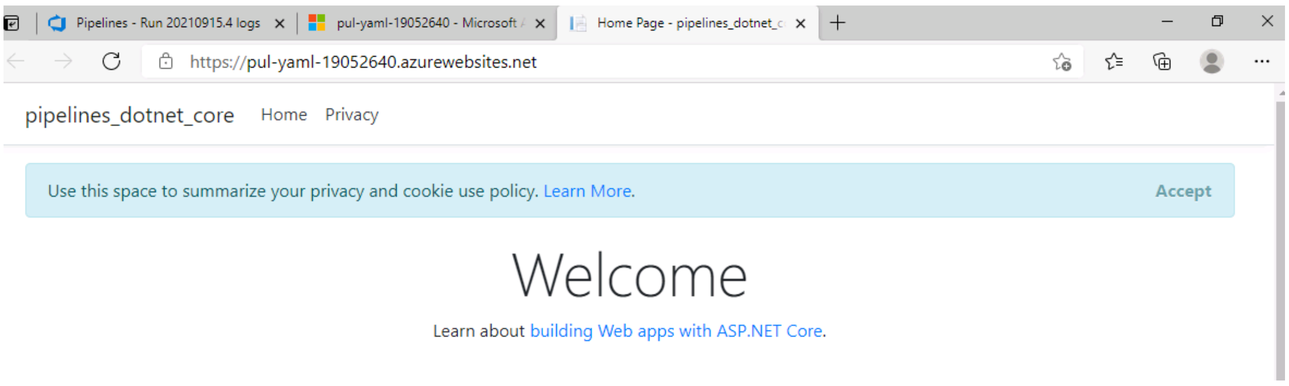


Resource group (change)
rg-lod19052640
Status
Running
Location
East US
Subscription (change)
URL
https://pul-yaml-19052640.azurewebsites.net
Health Check
Not Configured
App Service Plan
ASP-rglod19052640-bb1f (S1: 1)

19. Click on **Browse** or click on the URL for the App Service and you can see the app successfully running.



Resource group (change)
rg-lod19052640
Status
Running
Location
East US
Subscription (change)
URL
https://pul-yaml-19052640.azurewebsites.net
Health Check
Not Configured
App Service Plan
ASP-rglod19052640-bb1f (S1: 1)


Pipelines - Run 20210915.4 logs
pul-yaml-19052640 - Microsoft
Home Page - pipelines_dotnet_c
https://pul-yaml-19052640.azurewebsites.net
pipelines_dotnet_core
Home
Privacy
Accept

Use this space to summarize your privacy and cookie use policy. Learn More.

Welcome

Learn about building Web apps with ASP.NET Core.

> In this lab, you configured a self-hosted agent and created a YAML pipeline that builds an ASP.NET Core project on this self-hosted agent and deploys it to an Azure Web App.