Azure Developer Series

Migrating a dotnetcore 2-tier application to Azure, using different architectures and DevOps best practices

Hands-On-Labs step-by-step guides

Prepared by:

Peter De Tender

CEO and Lead Technical Trainer
PDTIT and 007FFFLearning.com

@pdtit @007FFFLearning

Version: Sept 2019 – 1.0

Contents

Migrating a dotnetcore 2-tiered application to Azure using different architectures and DevOps best practices - Hands-On-Labs step-by-step

You are part of an organization that is running a dotnetcore e-commerce platform application, using Windows Server infrastructure on-premises today, comprising a WebVM running Windows Server 2012 R2 with Internet Information Server (IIS) and a 2nd SQLVM running Windows Server 2012 R2 and SQL Server 2014.

The business has approved a migration of this business-critical workload to Azure, and you are nominated as the cloud solution architect for this project. No decision has been made yet on what the final architecture should or will look like. Your first task is building a Proof-of-Concept in your Azure environment, to test out the different architectures possible:

- Infrastructure as a Service (IAAS)
- Platform as a Service (PAAS)
- Containers as a Service (CaaS)

At the same time, your CIO wants to make use of this project to switch from a more traditional mode of operations, with barriers between IT sysadmin teams and Developer teams, to a DevOps way of working. Therefore, you are tasked to explore Azure DevOps and determine where CI/CD Pipelines can assist in optimizing the deployment and running operations of this e-commerce platform, especially when deploying updates to the application.

As you are new to the continuous changes in Azure, you want to make sure this process goes as smooth as possible, starting from the assessment to migration to day-to-day operations.

Abstract and Learning Objectives

This workshop enables anyone to learn, understand and build a Proof of Concept, in performing a multi-tiered .Net Core web application using Microsoft SQL Server database, platform migration to Azure public cloud, leveraging on different Azure Infrastructure as a Service, Azure Platform as a Service (PaaS) and Azure Container offerings like Azure Container Instance (ACI) and Azure Kubernetes Services (AKS).

After an introductory module on cloud app migration strategies and patterns, students get introduced to the basics of automating Azure resources deployments using Visual Studio and Azure Resource Manager (ARM) templates. Next, attendees learn about the importance of performing proper assessments, and what tools Microsoft offers to help in this migration preparation phase. Once the application has been deployed on Azure Virtual Machines, students learn about Microsoft SQL database migration to SQL Azure PaaS, as well as deploying and migrating web applications to Azure Web Apps.

After these foundational platform components, the workshop will totally focus on the core concepts and advantages of using containers for running business workloads, based on Docker, Azure Container Registry (ACR), Azure Container Instance (ACI) and WebApps for Containers, as well as how to enable container orchestration and cloud-scale using Azure Kubernetes Service (AKS).

In the last part of the workshop, students get introduced to Azure DevOps, the new Microsoft Application Lifecycle environment, helping in building a CI/CD Pipeline to publish workloads using the DevOps principals and concepts, showing the integration with the rest of the already touched on Azure services like Azure Web Apps and Azure Kubernetes Services (AKS), closing the workshop with a module on overall Azure monitoring and operations and what tools Azure has available to assist your IT teams in this challenge.

The focus of the workshop is having a Hands-On-Labs experience, by going through the following exercises and tasks:

- Deploying a 2-tier Azure Virtual Machine (Webserver and SQL database Server) using ARM-template automation with Visual Studio 2019;
- Publishing a .NET Core e-commerce application to an Azure Web Virtual Machine and SQL DB Virtual Machine;
- Performing a proper assessment of the as-is Web and SQL infrastructure using Microsoft Assessment Tools;
- Migrating a SQL 2014 database to Azure SQL PaaS (Lift & Shift);
- Migrating a .NET Core web application to Azure Web Apps (Lift & Shift);
- Containerizing a .NET Core web application using Docker, and pushing to Azure Container Registry (ACR);
- Running Azure Container Instance (ACI) and WebApp for Containers;
- Deploy and run Azure Azure Kubernetes Services (AKS);

- Deploying Azure DevOps and building a CI/CD Pipeline for the subject e-commerce application;
- Managing and Monitoring Azure Kubernetes Services (AKS);

Requirements

Naming Conventions:

IMPORTANT: Most Azure resources require unique names. Throughout these steps you will see the word **"[SUFFIX]"** as part of resource names. You should replace this with your initials, guaranteeing those resources get uniquely named.

Azure Subscription:

Participants need a "pay-as-you-go", MSDN or other paid Azure subscription

a) In one of the Azure Container Services tasks, you are required to create an Azure AD Service Principal, wich typically requires an Azure subscription owner to log in to create this object. If you don't have the owner right in your Azure subscription, you could ask another person to execute this step for you.
b) The Azure subscription must allow you to run enough cores, used by the baseline Virtual Machines, but also later on in the tasks when deploying the Azure Container Services, where ACS agent and master machines are getting set up. If you follow the instructions as written out in the lab guide, you need 12 cores.
c) If you run this lab setup in your personal or corporate Azure payable subscription, using the configuration as described in the lab guide, the estimated Azure consumption costs for running the setups during the 2 days of the workshop is $20.

Other requirements:

Participants need a local client machine, running a recent Operating System, allowing them to:

- browse to https://portal.azure.com from a most-recent browser;
- establish a secured Remote Desktop (RDP) session to a lab-jumpVM running Windows Server 2016;

Alternative Approach:

Where the lab scenario assumes all exercises will be performed from within the lab-jumpVM, (since several tools will be installed on the lab-jumpVM or are already installed by default), participants could also execute (most, if not all…) steps from their local client machine.

The following tools are being used throughout the lab exercises:

- Visual Studio 2017 community edition (updated to latest version); this could also be Visual Studio 2019 community edition - latest version
- Docker for Windows (updated to latest version)
- Azure CLI 2.0 (updated to latest version)
- Kubernetes CLI (updated to latest version)
- SimplCommerce Open Source e-commerce platform example (http://www.simplcommerce.com)

Make sure you have these tools installed prior to the workshop, if you are not using the lab-jumpVM. You should also have full administrator rights on your machine to execute certain steps within using these tools.

Final Remarks:

VERY IMPORTANT: You should be typing all of the commands as they appear in the guide, except where explicitly stated in this document. Do not try to copy and paste from Word to your command windows or other documents where you are instructed to enter information shown in this document. There can be issues with Copy and Paste from Word or PDF that result in errors, execution of instructions, or creation of file content.

IMPORTANT: Most Azure resources require unique names. Throughout these steps you will see the word "[SUFFIX]" as part of resource names. You should replace this with your initials, guaranteeing those resources get uniquely named.

Lab 8: Deploying Azure Workloads using Azure DevOps

What you will learn

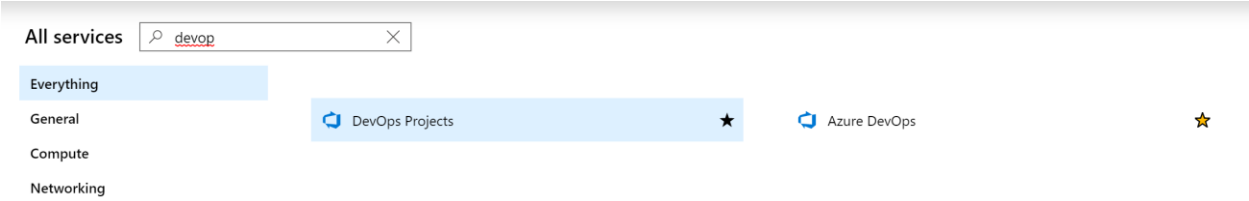
In this next lab of this workshop, you get introduced to Azure DevOps, how to deploy the service as part of your Azure subscription, as a starting point. Next, you will learn the concepts of building a CI/CD Pipeline, to deploy Azure workloads. Starting from the source code of our sample e-commerce application in a GitHub repo, you will also learn how to deploy the previously built Docker container and run this in Azure Container Instance, but deployed using Azure DevOps. Lastly, you will deploy the Docker container to the AKS Cluster you deployed earlier, again using Azure DevOps.

Time Estimate

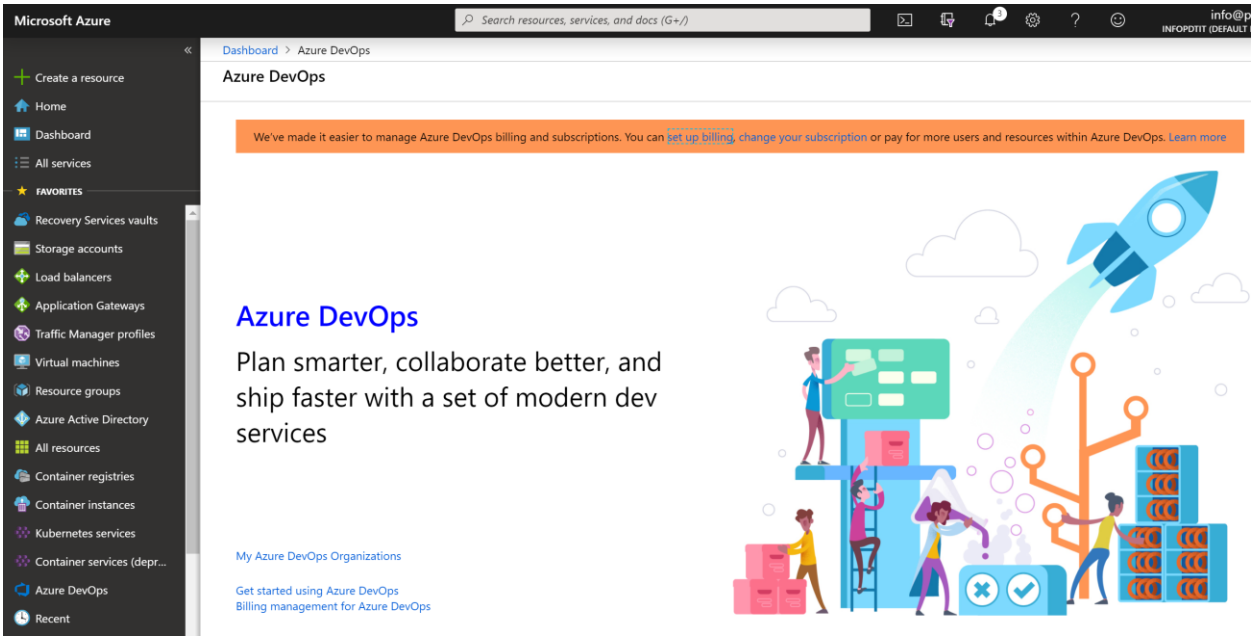
This lab shouldn't take longer than 60 minutes.

Task 1: Deploying an Azure DevOps Organization

1. From your Azure Portal, **browse to all services**, and search for **DevOps.**

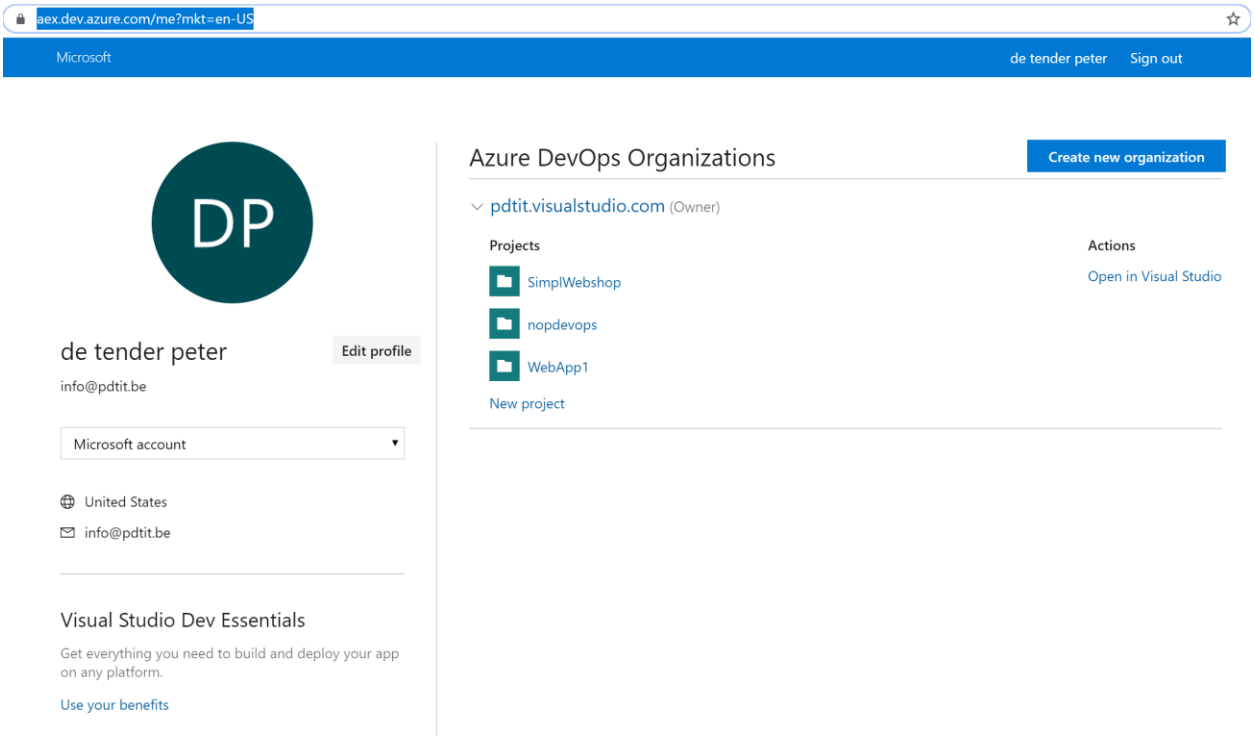


2. Select "Azure DevOps"

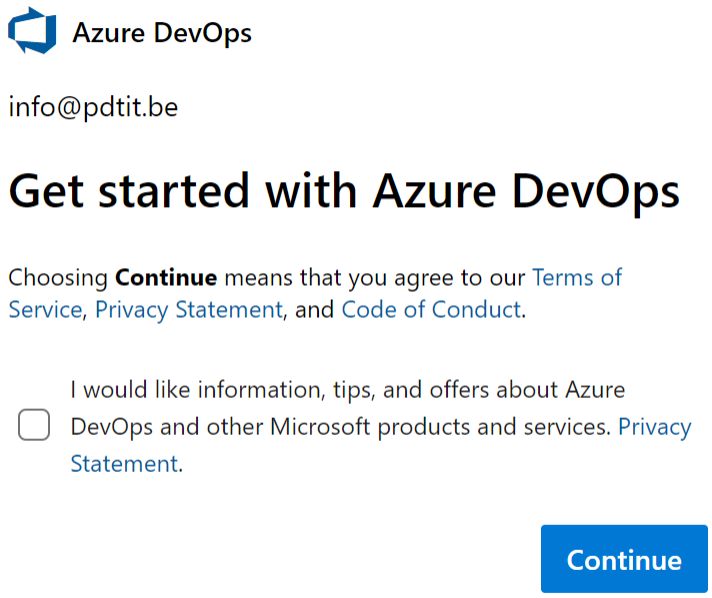


3. Next, click "My Azure DevOps Organizations"

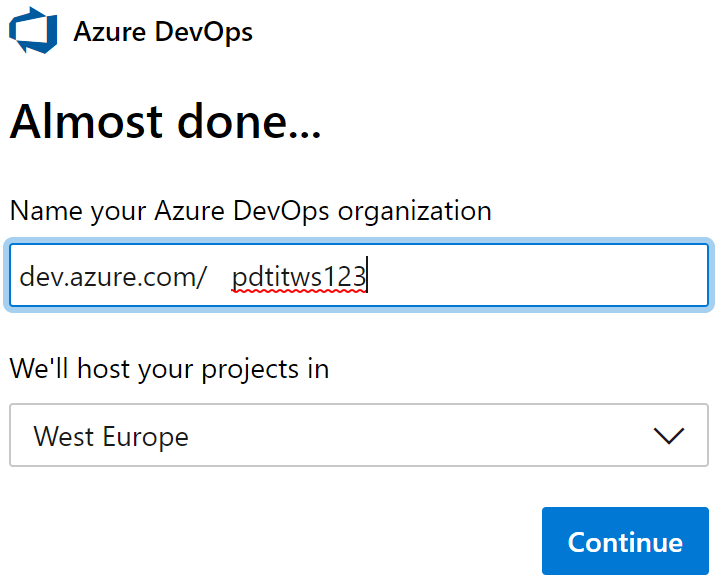
After providing your Azure credentials once more, you will get redirected to a visualstudio.com website, and eventually ending up at https://aex.dev.azure.com/me?mkt=en-US



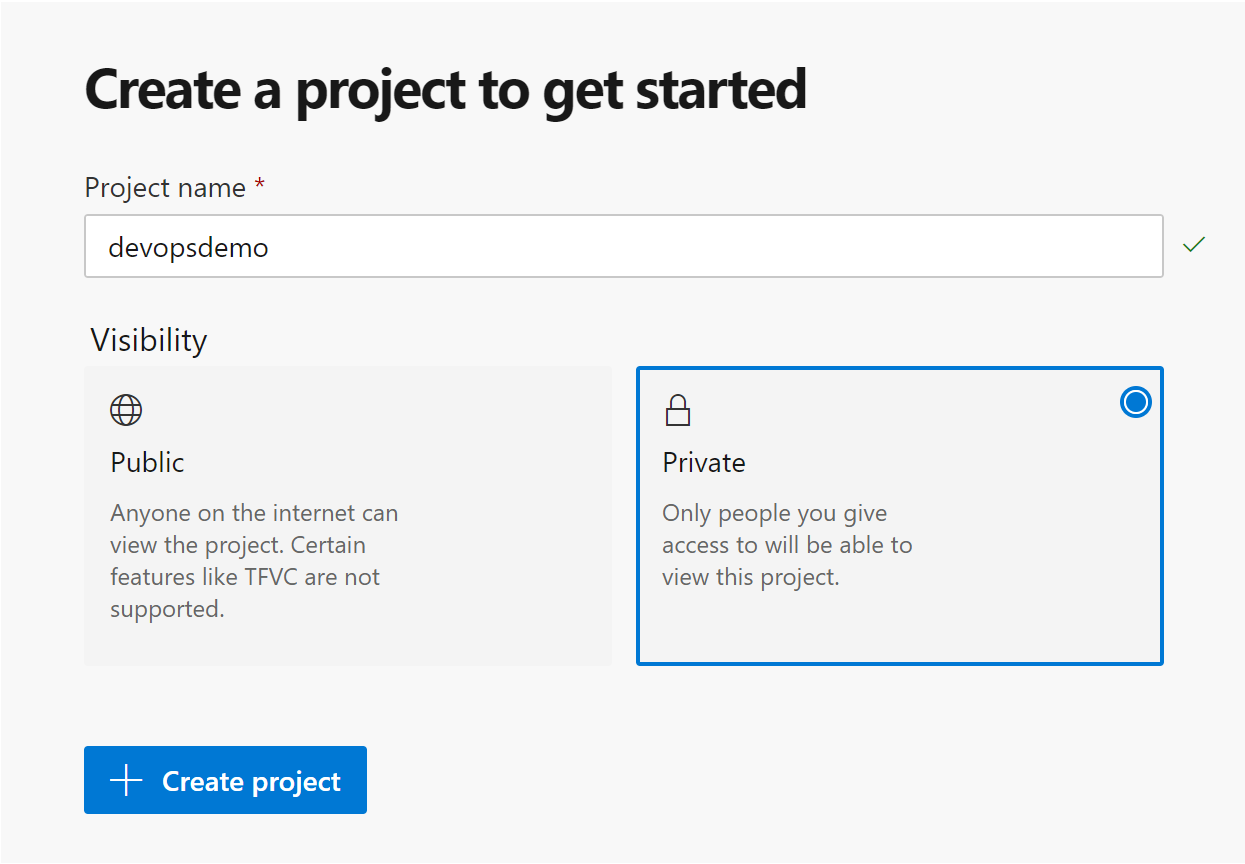
4. Here, select “Create New Organization”



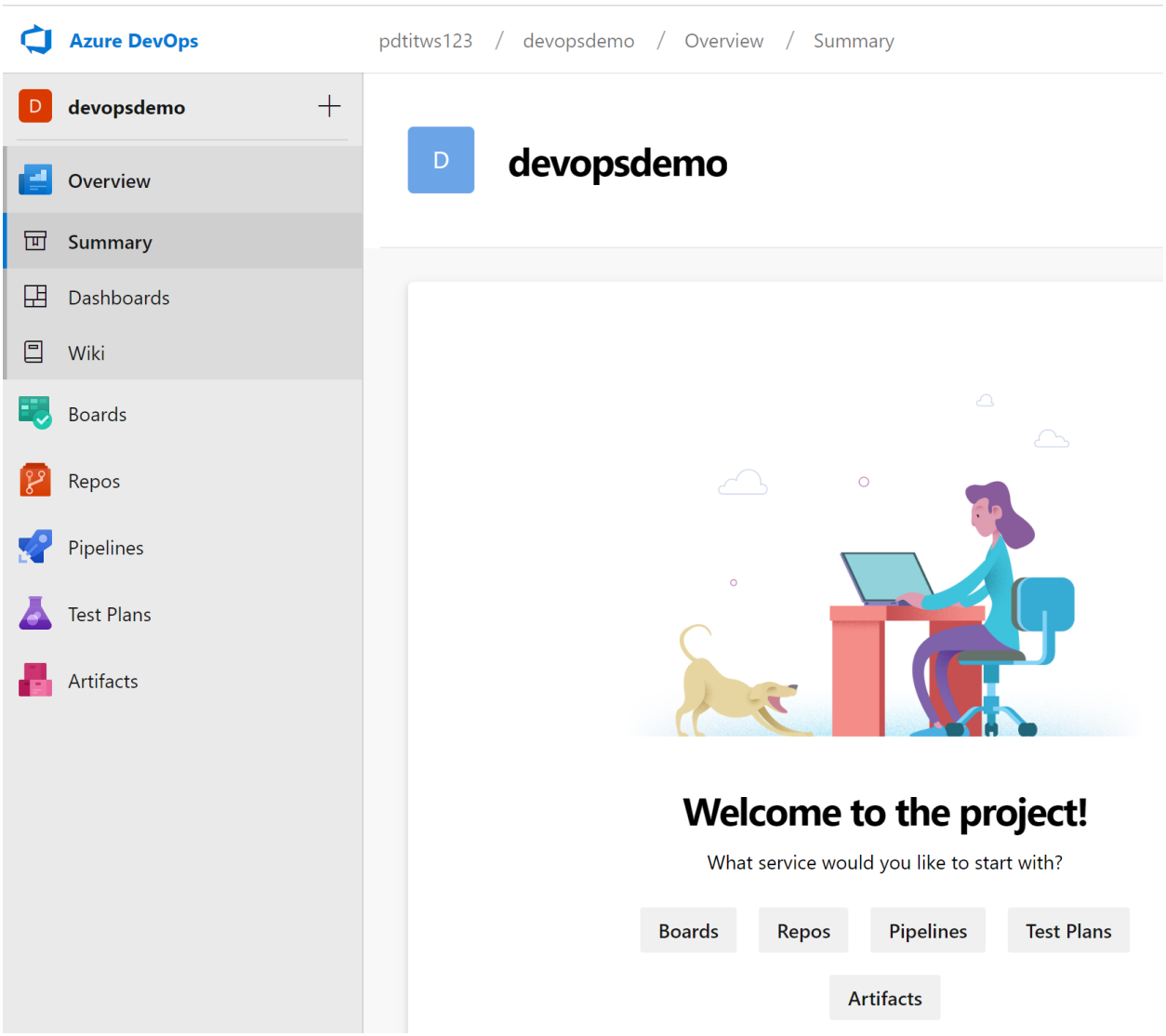
5. Press Continue



6. Press Continue, and provide a name for your DevOps Project



7. Confirm by **clicking the "+ Create Project"** button; your Azure DevOps "Workspace" gets created

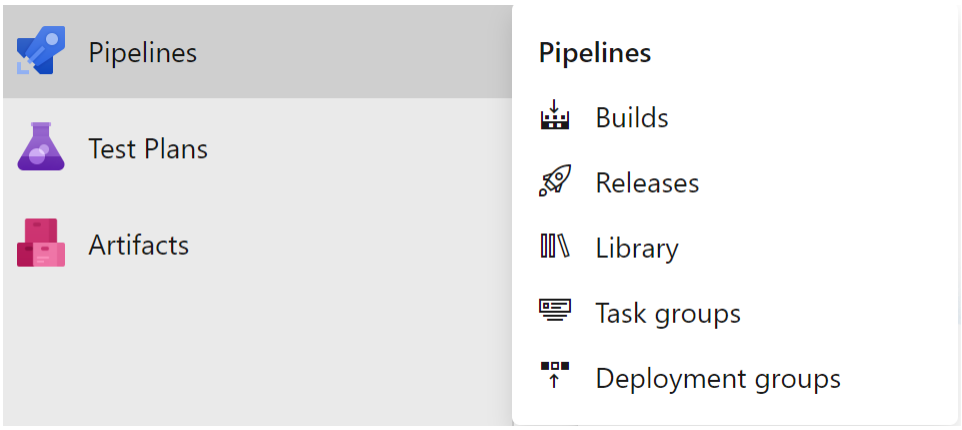


8. This completes the task, in which you deployed Azure DevOps and configured an Azure DevOps Organization. In the next task, you will build your DevOps Pipelines.

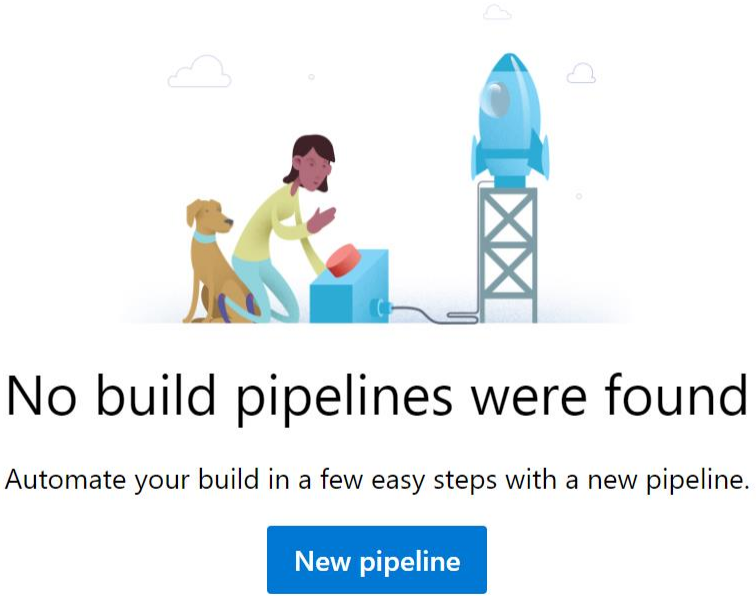
Task 2: Creating and Deploying an Azure Build Pipeline for your application

While Azure DevOps gives you an end-to-end solution to manage your application development and deployment lifecycle, this lab focuses mainly on the **Azure Pipelines** service within.

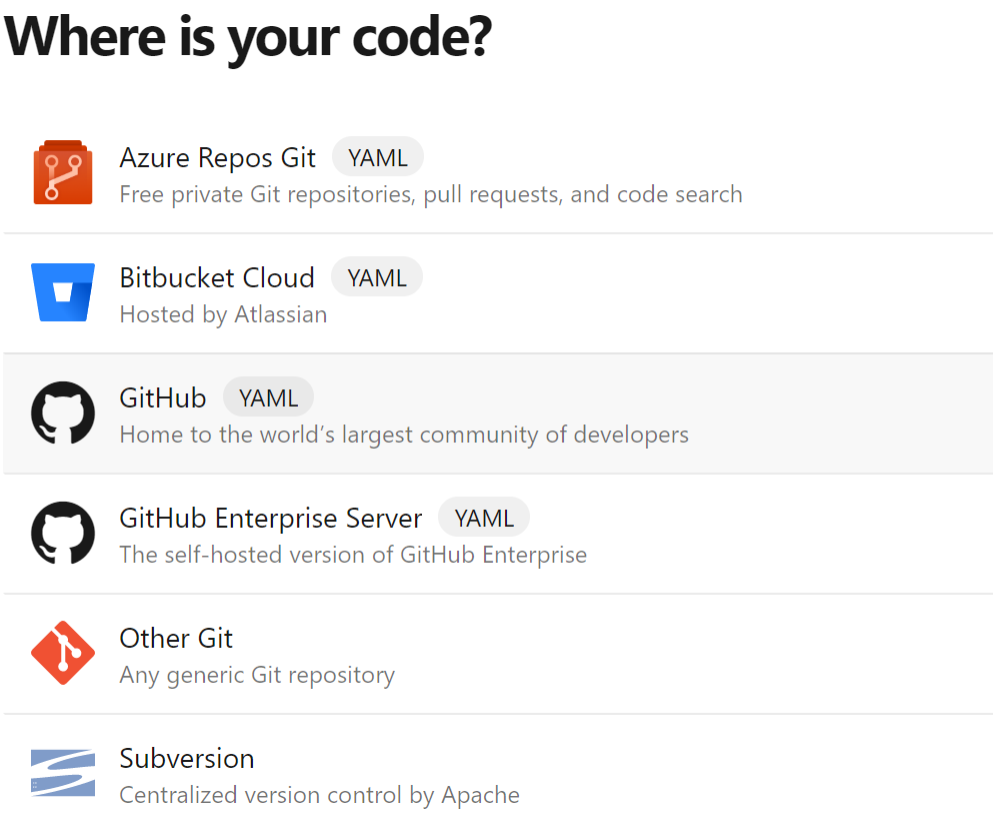
1. Select "Pipelines"



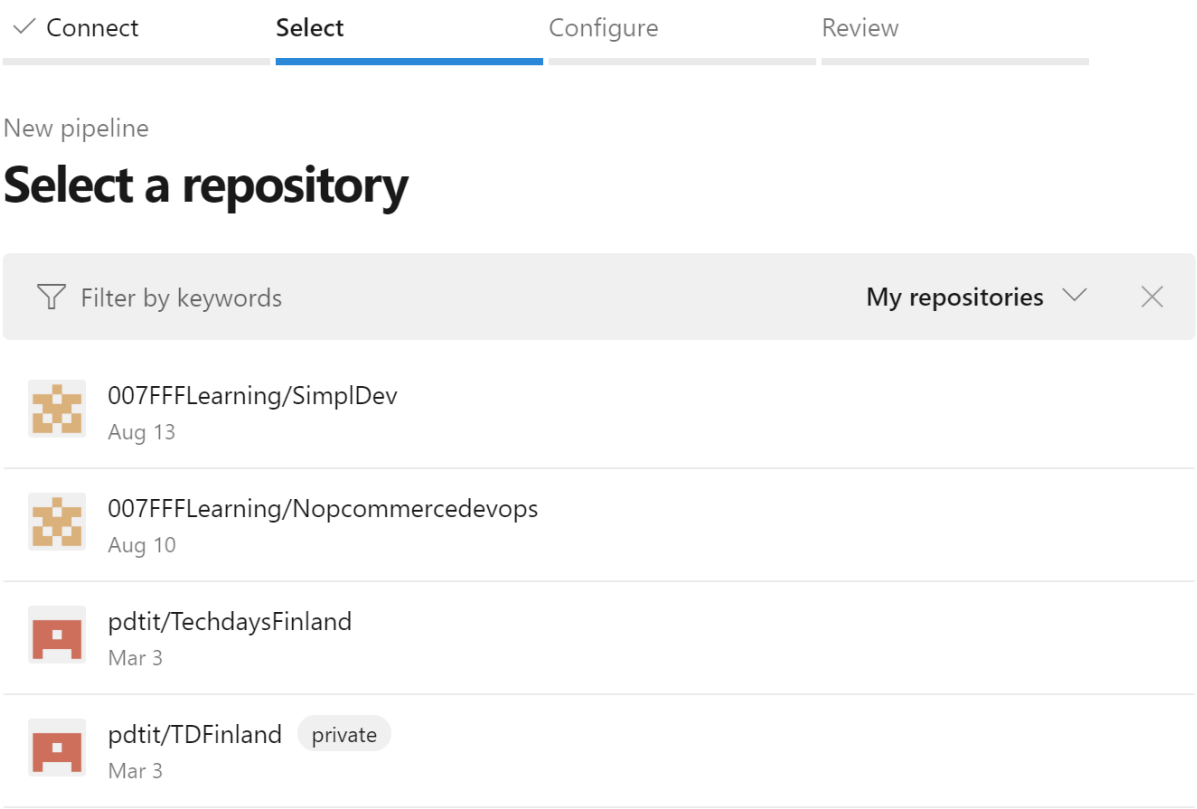
2. Select "**Builds**"



3. Click "New Pipeline"



4. You are prompted for selecting your version control repository. Select GitHub



Note: you will be prompted to create an integration between Azure DevOps and Github, by authentication with your GitHub account. If you don't have one yet, go to http://www.github.com, and create one.

Once you have your GitHub account ready, you can create a new GitHub repo, and upload the SimplCommerce source files into it. Another option is cloning the source files from the sample repo I am using here 007FFFLearning/SimplDev

check the GitHub website for instructions on how to clone an existing repo into your own GitHub.

5. In the next step, you are presented with an Azure Devops Pipeline.yml file; as you know, this file is automatically being build up, based on the detection of the source code language.

✓ Connect ✓ Select ✓ Configure **Review**

New pipeline

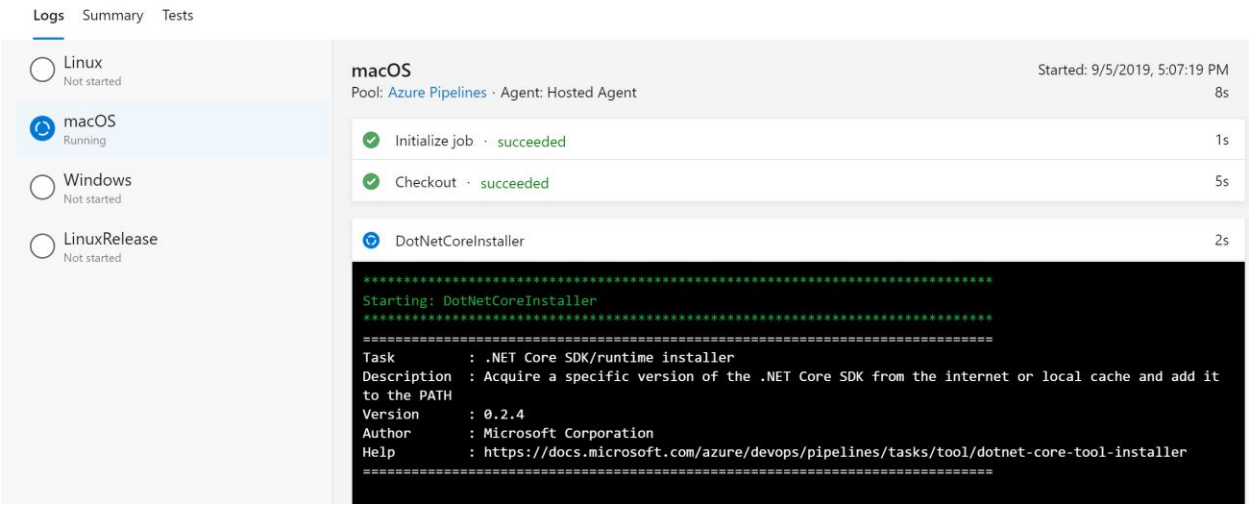
Review your pipeline YAML

azure-pipelines.yml

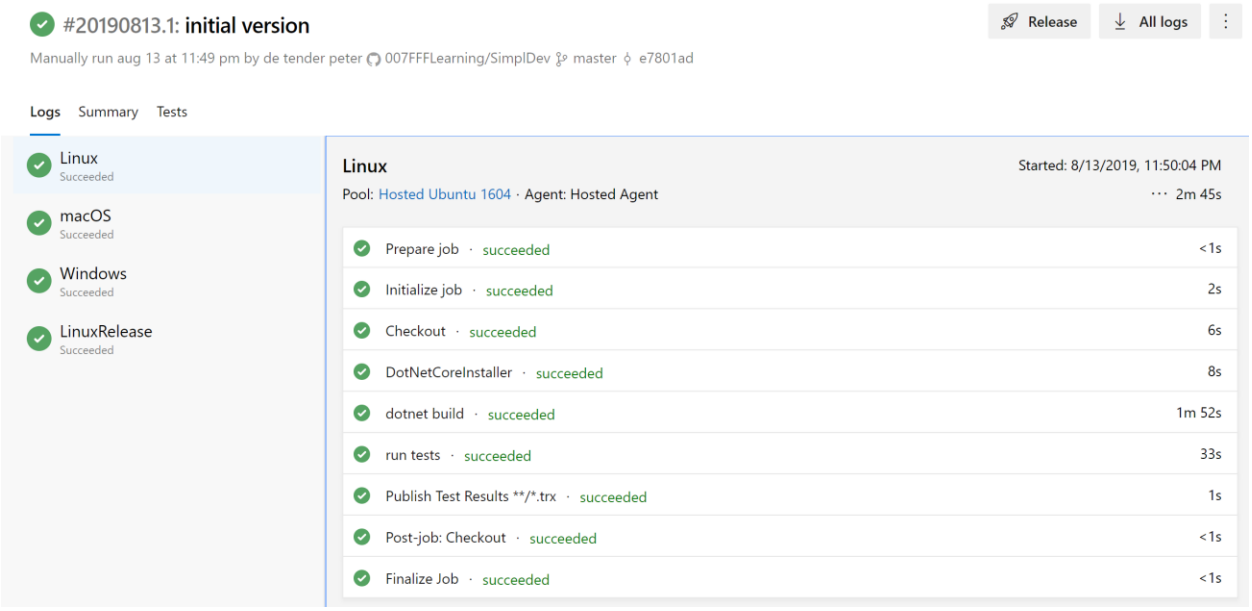
```yaml
# ASP.NET Core
# Build and test ASP.NET Core projects targeting .NET Core.
# Add steps that run tests, create a NuGet package, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/dotnet-core

trigger:
- master

jobs:
- job: Linux
  pool:
    vmImage: 'Ubuntu 16.04'
  steps:
  - task: DotNetCoreInstaller@0
    inputs:
      packageType: 'sdk'
      version: '2.2.401'
  - script: dotnet build ./SimplCommerce.sln
    displayName: 'dotnet build'
  - script: ./run-tests.sh
    displayName: 'run tests'
  - task: PublishTestResults@2
    displayName: 'Publish Test Results **/*.trx'
    condition: succeededOrFailed()
    inputs:
      testResultsFormat: VSTest
      testResultsFiles: '**/*.trx'
```

6. For example here, it recognizes the source code as **.NET Code**, and will create several build pipelines, for different platforms (Linux, MacOS, Windows). If you want, you could remove parts of this pipeline file, and only testing against Linux OS for example.

7. Confirm by clicking **Run;** your pipeline build process is starting



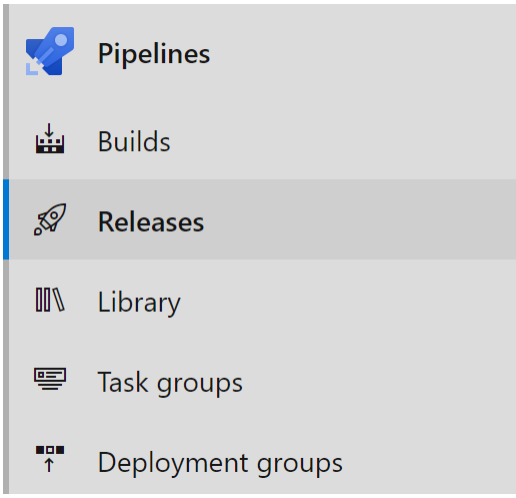
8. Wait for this process to complete successfully.



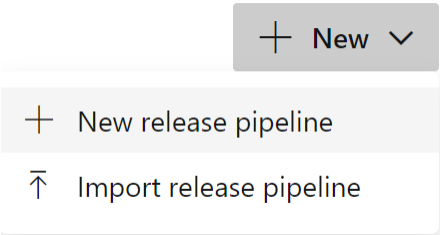
9. This completes the task in which you set up a Build Pipeline, based on application source code in GitHub. In the next task, we will continue the process, by creating and running a release pipeline, publishing the code to Azure.

Task 3: Building a Release Pipeline in Azure DevOps

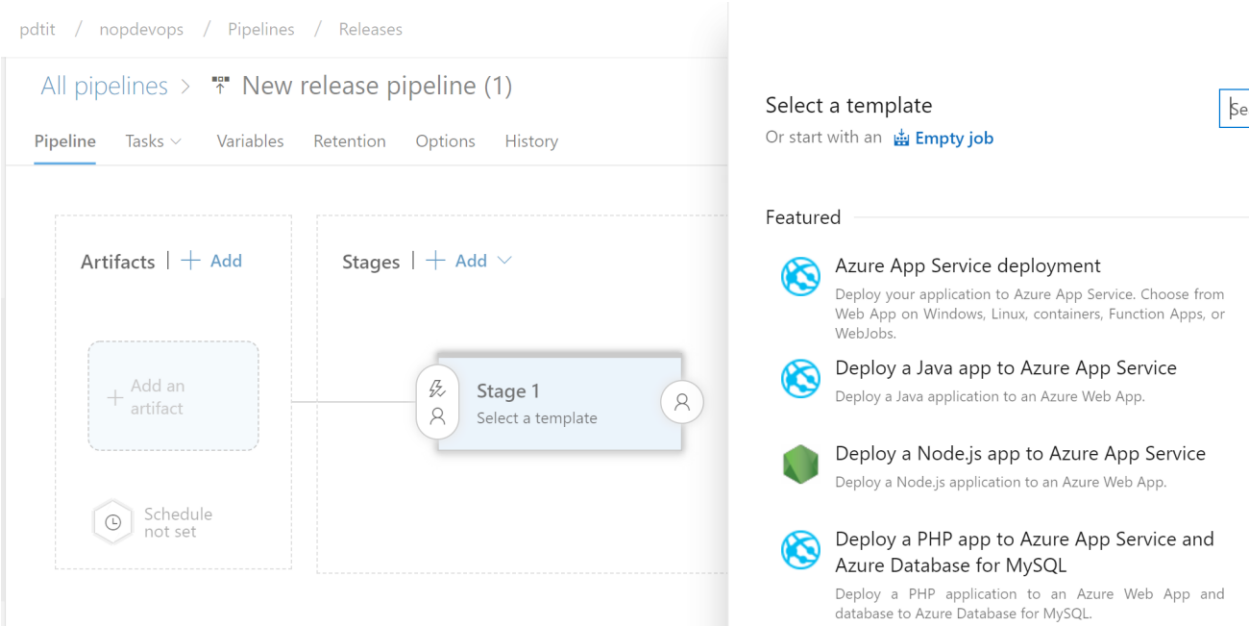
1. From Azure DevOps, Pipelines, select **Releases**



2. Next, select **New / Release Pipeline**

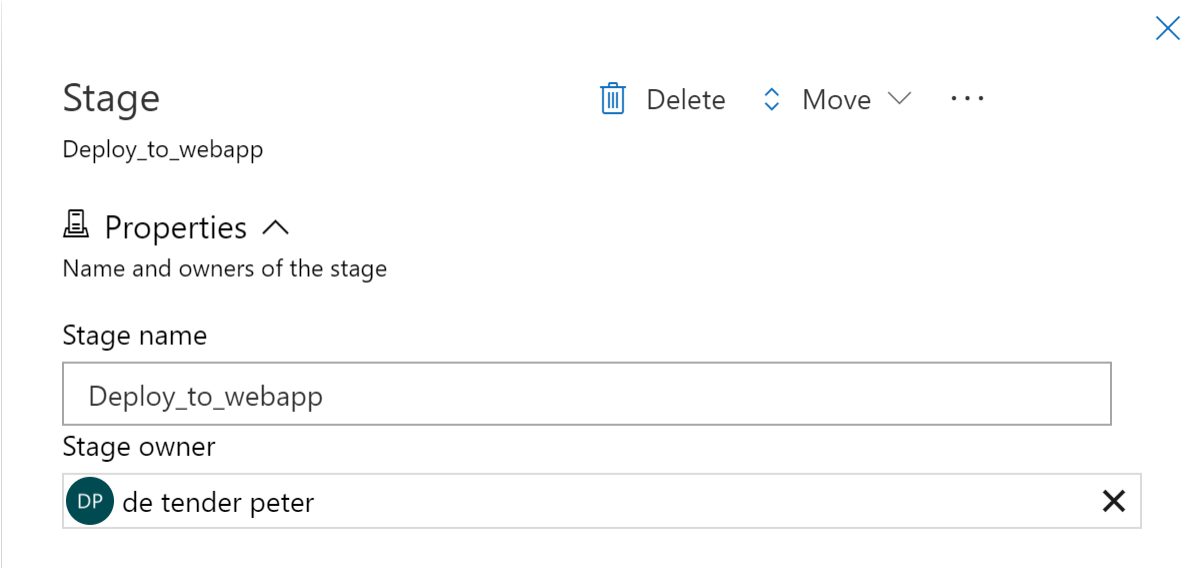


3. This launches the New release pipeline creating wizard.

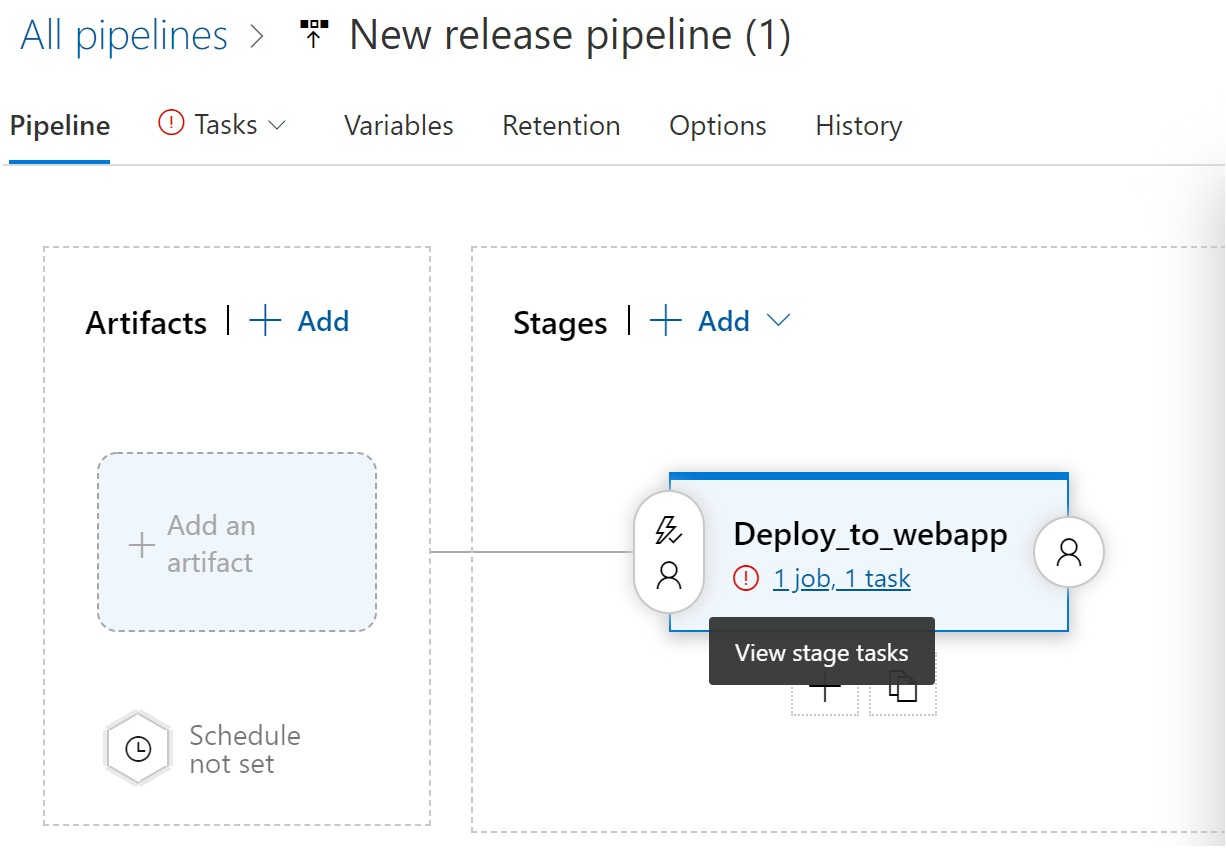


4. From the **Template list**, select **Azure App Service Deployment**

   Provide a description for the **Stage Name, for example Deploy_to_webapp**



5. The Pipeline now looks like this:



6. In the **Stage** field, select "**1 job, 1 task**"; this is where you will provide the settings of the Azure Web App environment you will use for the actual deployment.

Stage name

Deploy_to_webapp

Parameters | Unlink all

Azure subscription * | Manage

This setting is required.

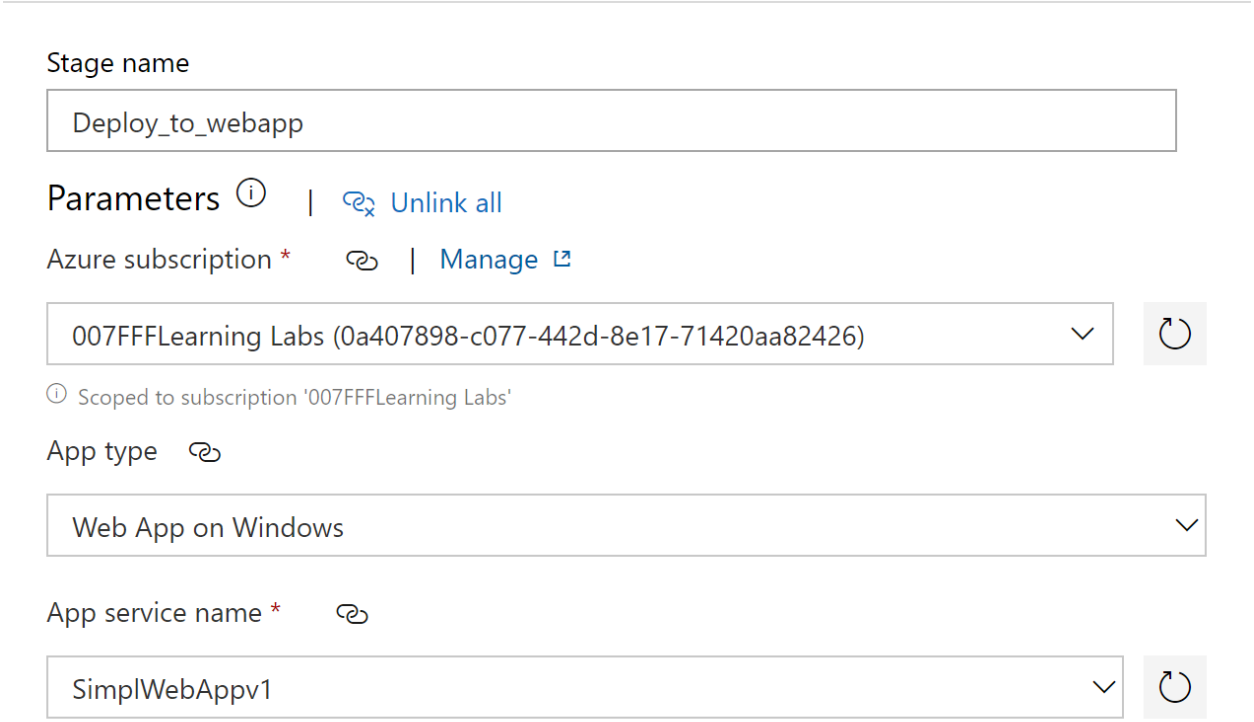
App type

Web App on Windows

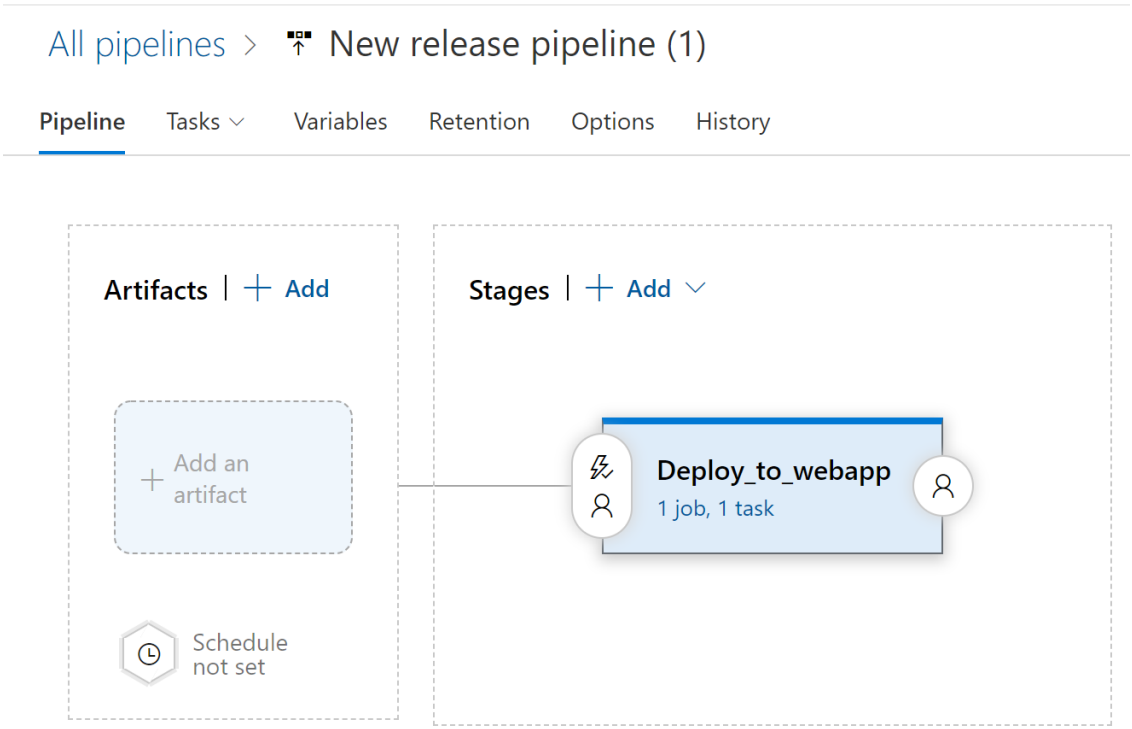
App service name *

This setting is required.

7. Complete the settings according to your Azure subscription and resources you already have from earlier deployments.

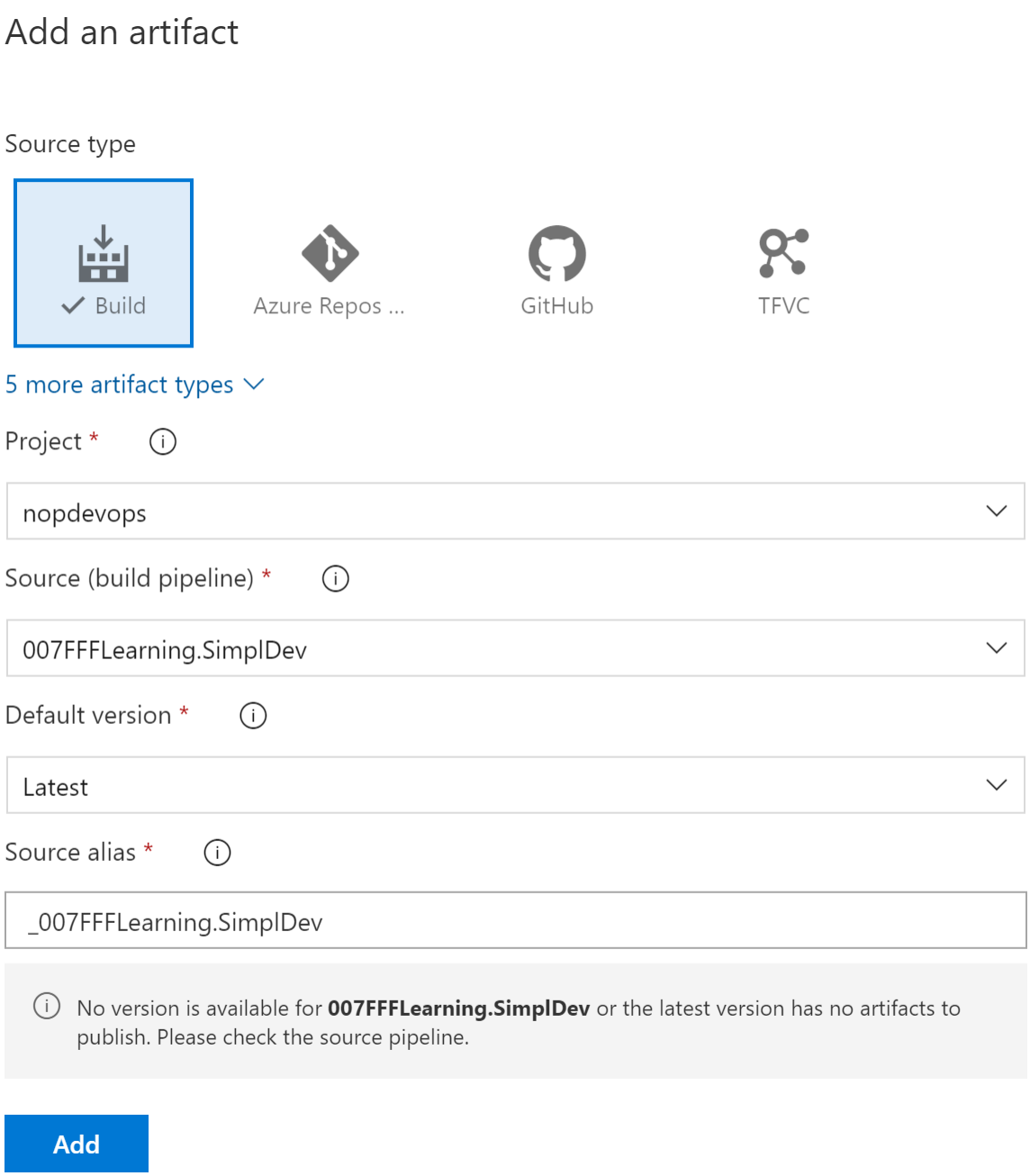


8. When done, **click Pipeline** in the top menu of your Azure Pipeline project, **to return and complete the next step in the Pipeline process, specifying the Artifacts**.



9. Select Artifacts / Add an Artifact

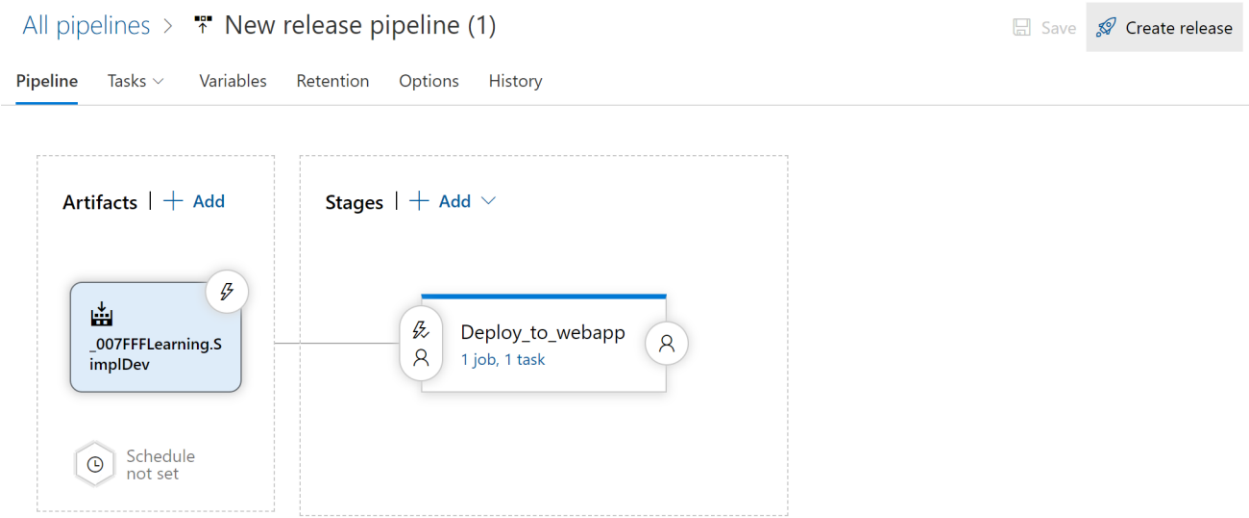
10. Complete the parameters, according to the Build Pipeline we created in the previous task:



11. Confirm by clicking "Add"

12. When done, click the "Save" button

13. Next, click the “Create Release” button



14. Save the changes, and confirm with “Create”

Create a new release

New release pipeline (1)

Pipeline

Click on a stage to change its trigger from automated to manual.

Deploy_to_we

Stages for a trigger change from automated to manual.

Deploy_to_webapp

Artifacts

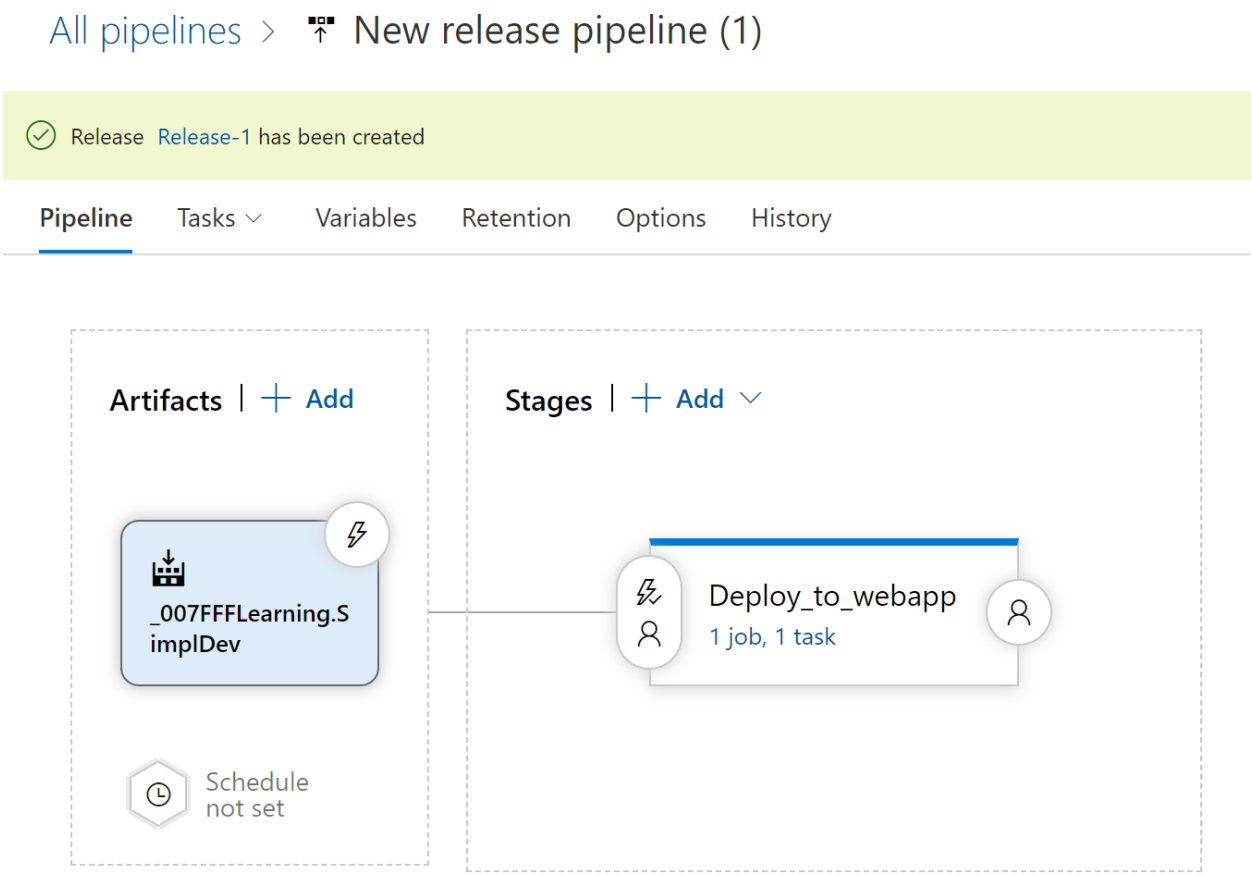
Select the version for the artifact sources for this release

Source alias	Version
_007FFFLearning.SimplDev	20190813.1

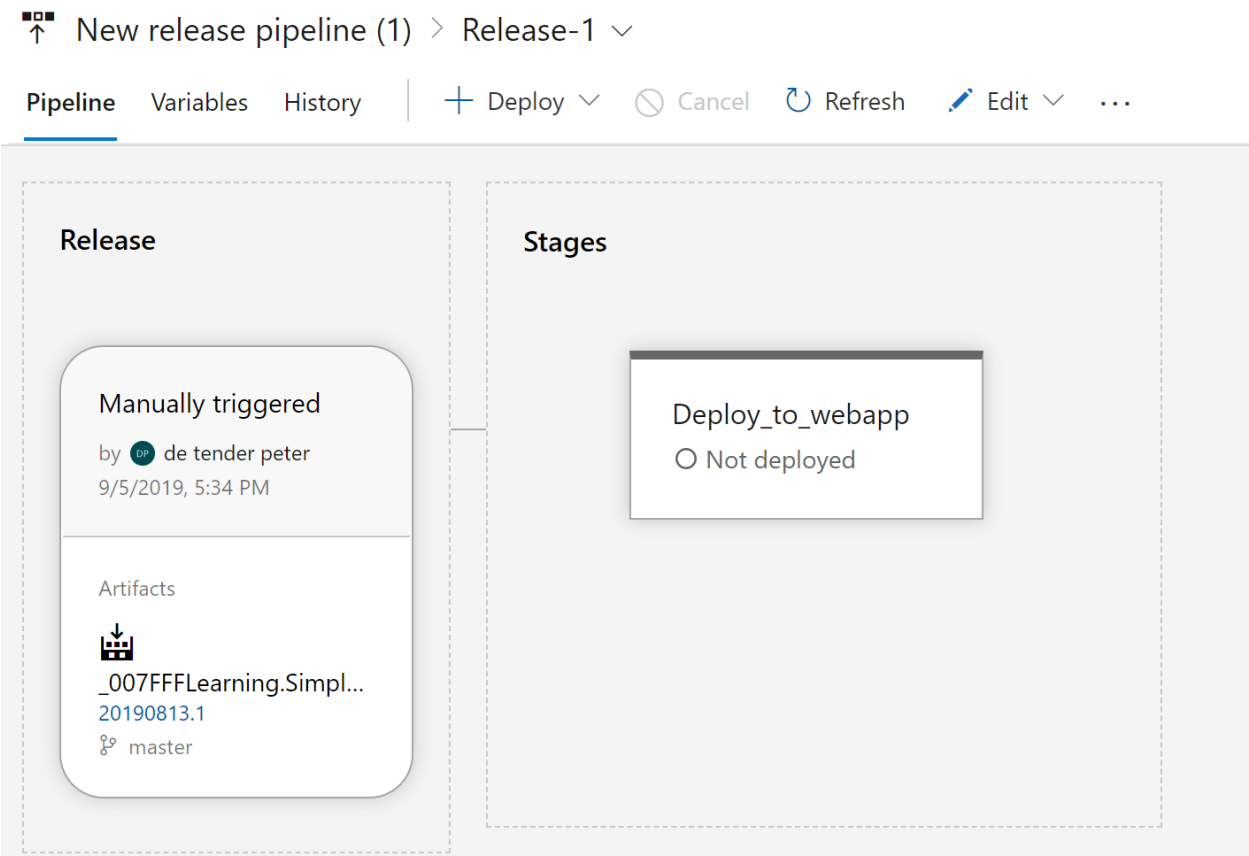
Release description

Create Cancel

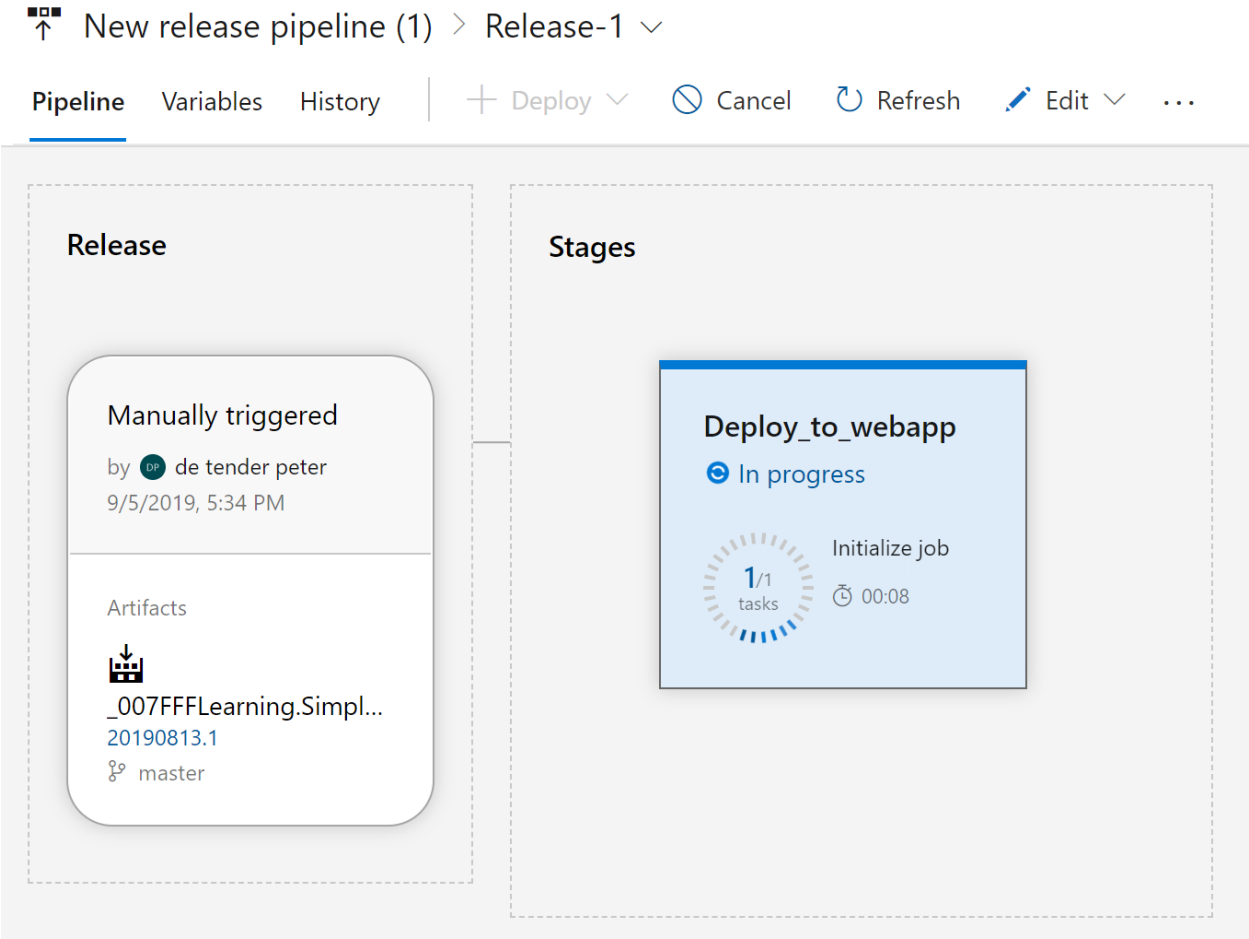
15. You get prompted the release is getting created



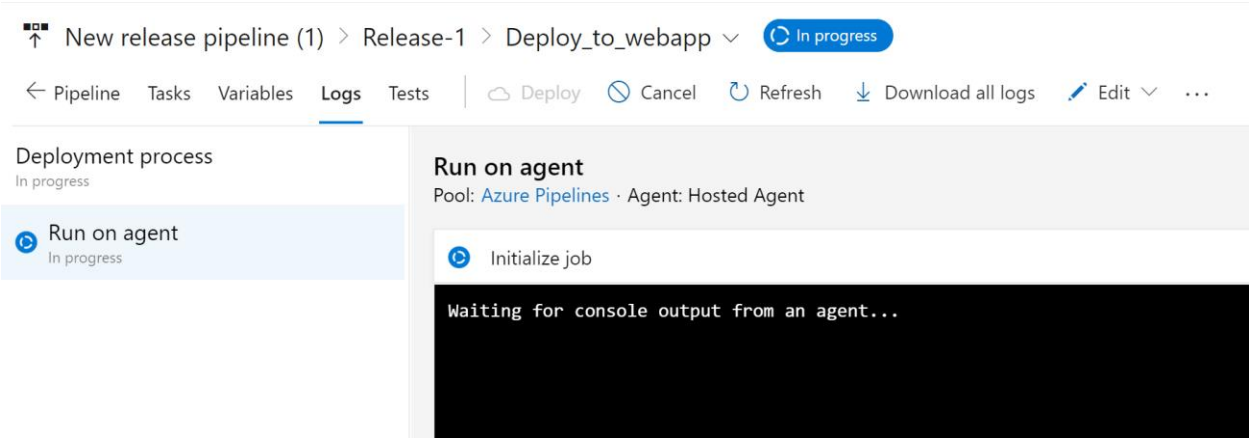
16. Click "Release-1", which redirects you to the details of the Release process



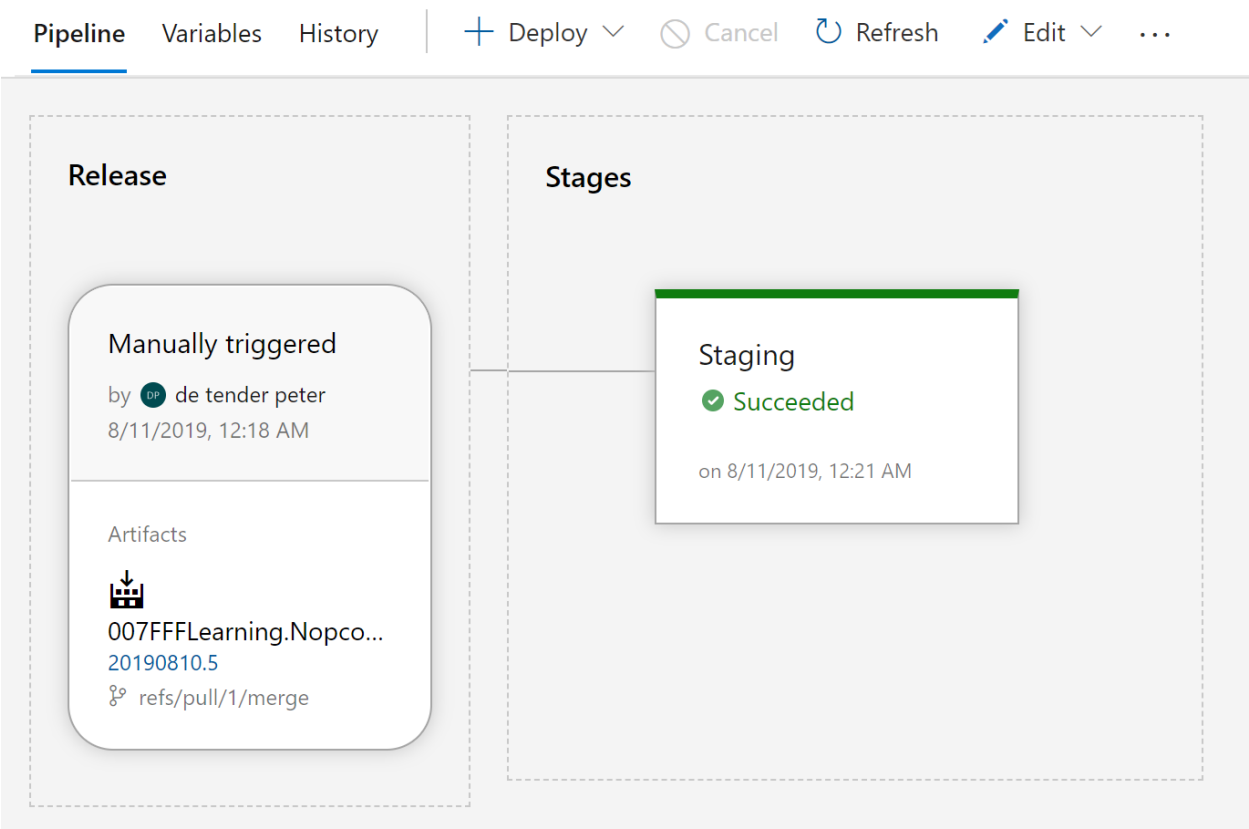
17. Select "Deploy_to_webapp", and confirm "Deploy"; the status will change to "in progress"



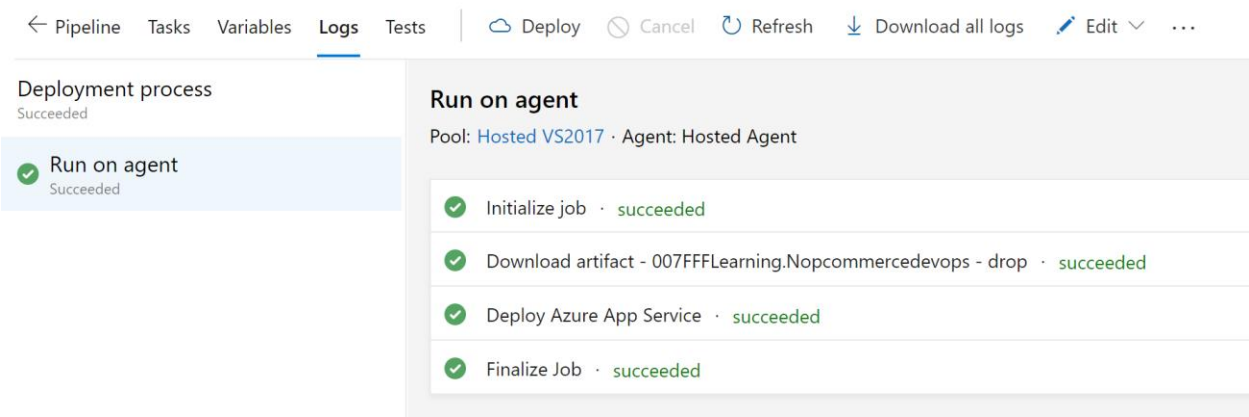
18. Wait for this process to initialize



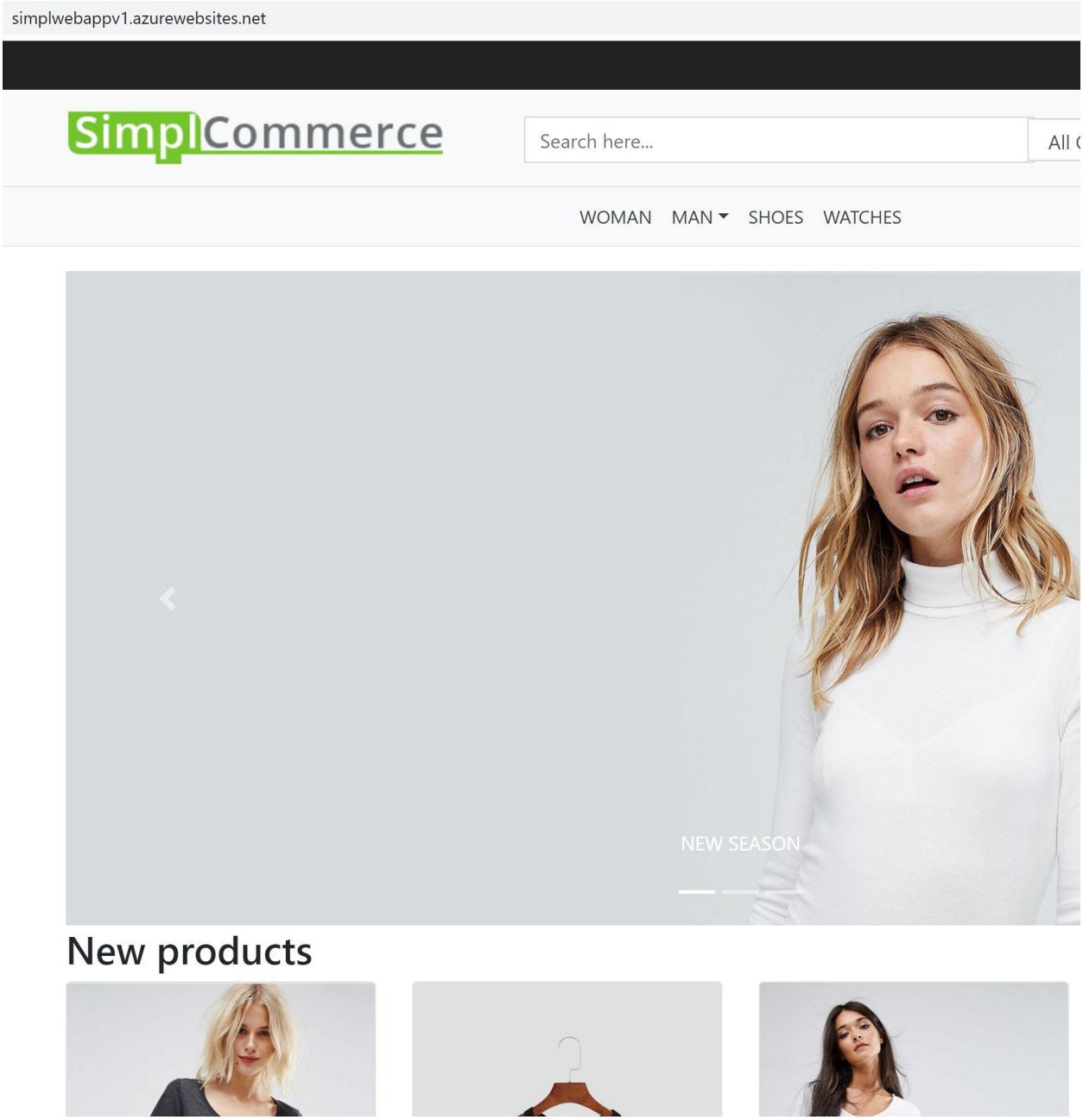
19. Wait for the release stage to complete successfully



20. Selecting the **Staging**, will open the details of the release process



21. From the Azure Portal, browse to the **Azure WebApp** you selected in the **Release Pipeline** as target, and validate it is running as expected.



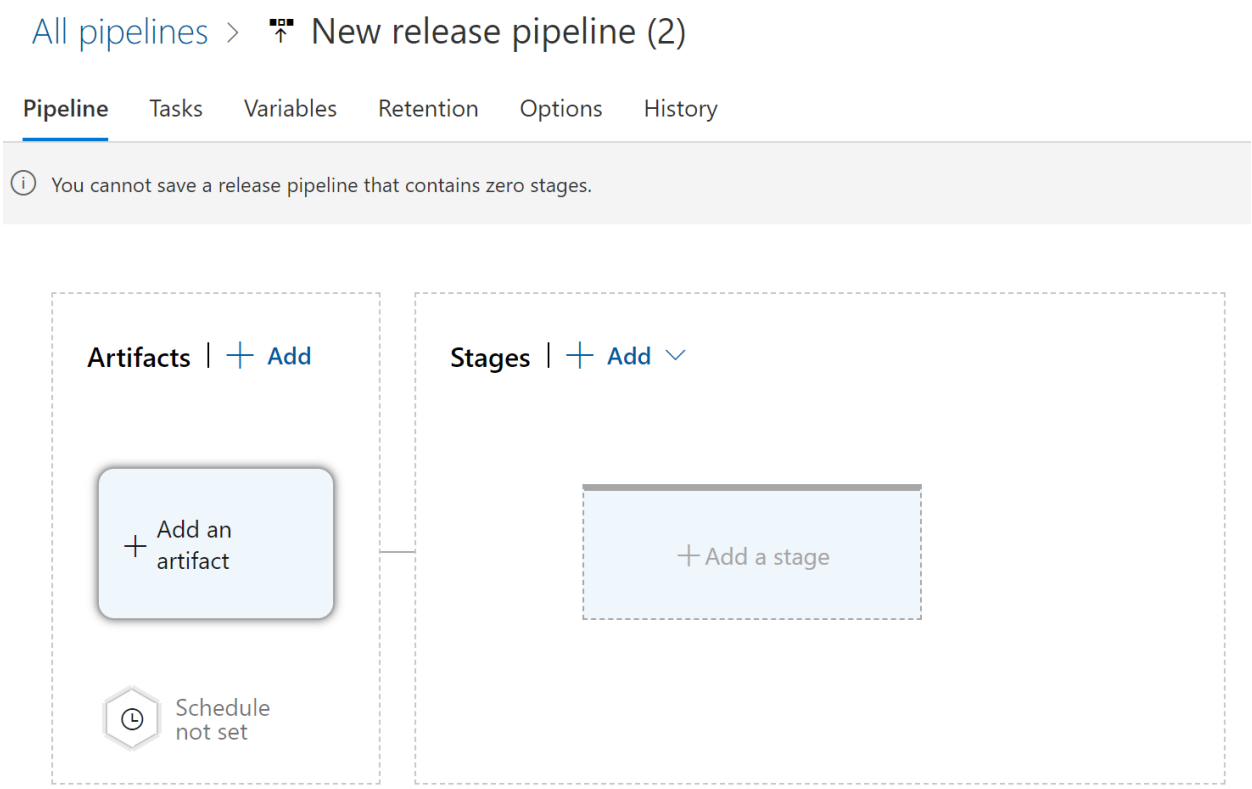
22. This completes the task in which you created a Release Pipeline, based on a previous Build Pipeline configuration.

Task 4: Creating a Release Pipeline for Docker Containers from ACR

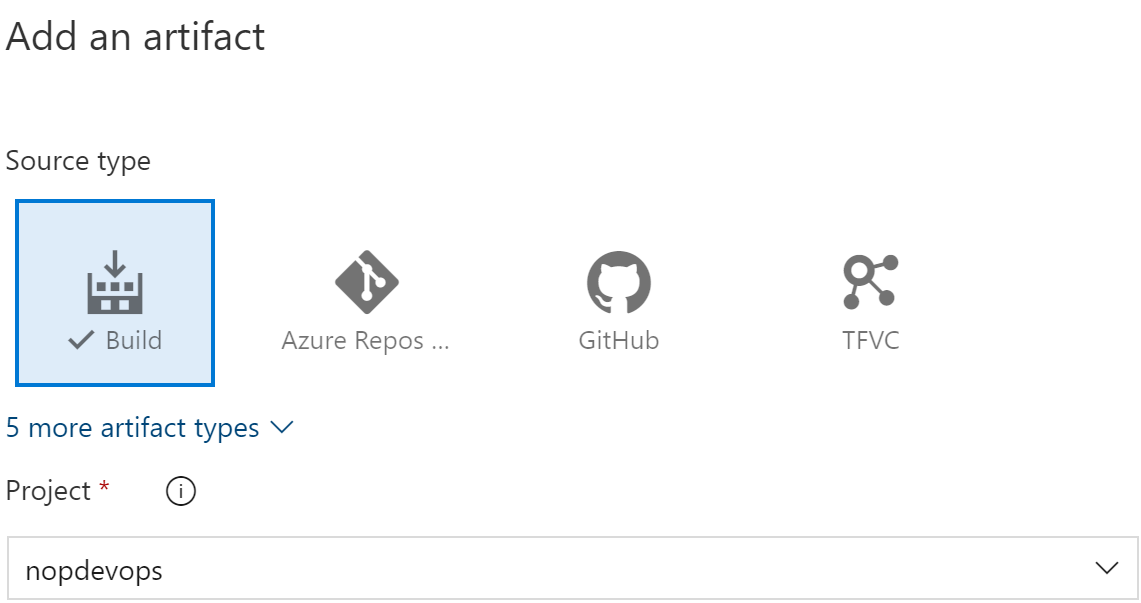
Similar to the previous Release Pipeline from source code in GitHub to a published Azure WebApp, we can use the same concept to create a release pipeline, based on a Docker container in Azure Container Registry. This is similar to the manual task you ran in lab 4 earlier.

1. From **Azure DevOps**, select **Pipeline / Release / New Release Pipeline**

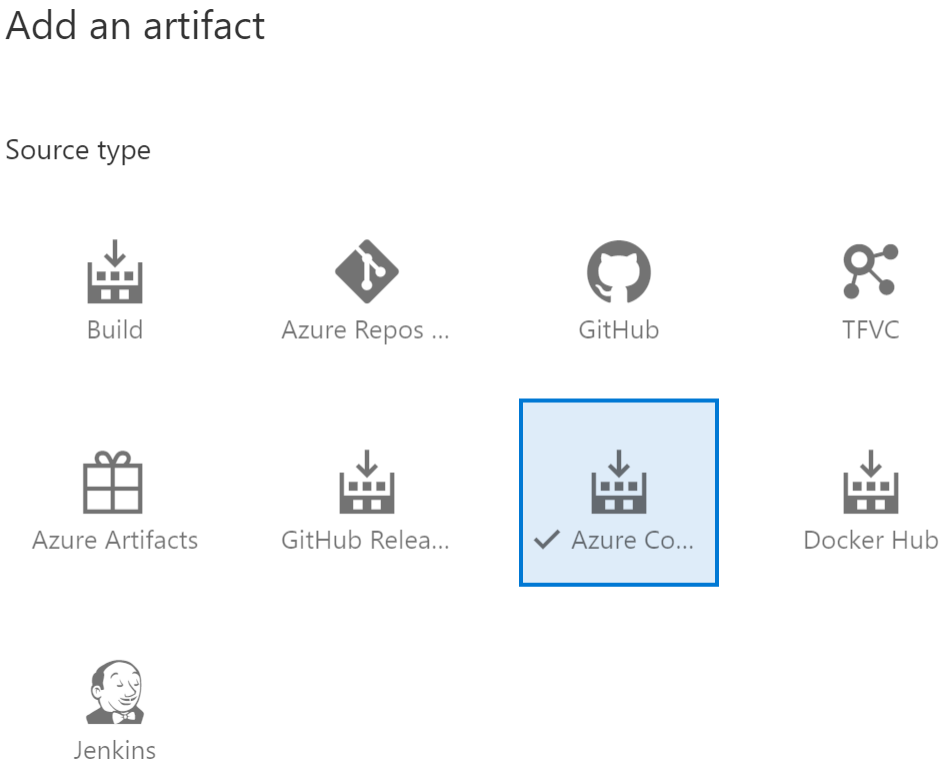
2. **When the Template window appears, close it, and select** Artifacts first



3. Choose "**Add an Artifact"; From the Artifact blade, click "5 more artifact types",** to extend the list of artifacts to choose from.



4. Select "Azure Container Registry", and complete the parameters according to the existing resources in your Azure subscription, reusing the resources from previous lab exercises (Azure Container Registry, Repository,...)



Service connection * | Manage

007FFFLearning Labs (0a407898-c077-442d-8e17-71420aa82426)

Resource Group *

noprg

Azure Container Registry *

nopacr1

Repository *

simplpdtv1

Default version *

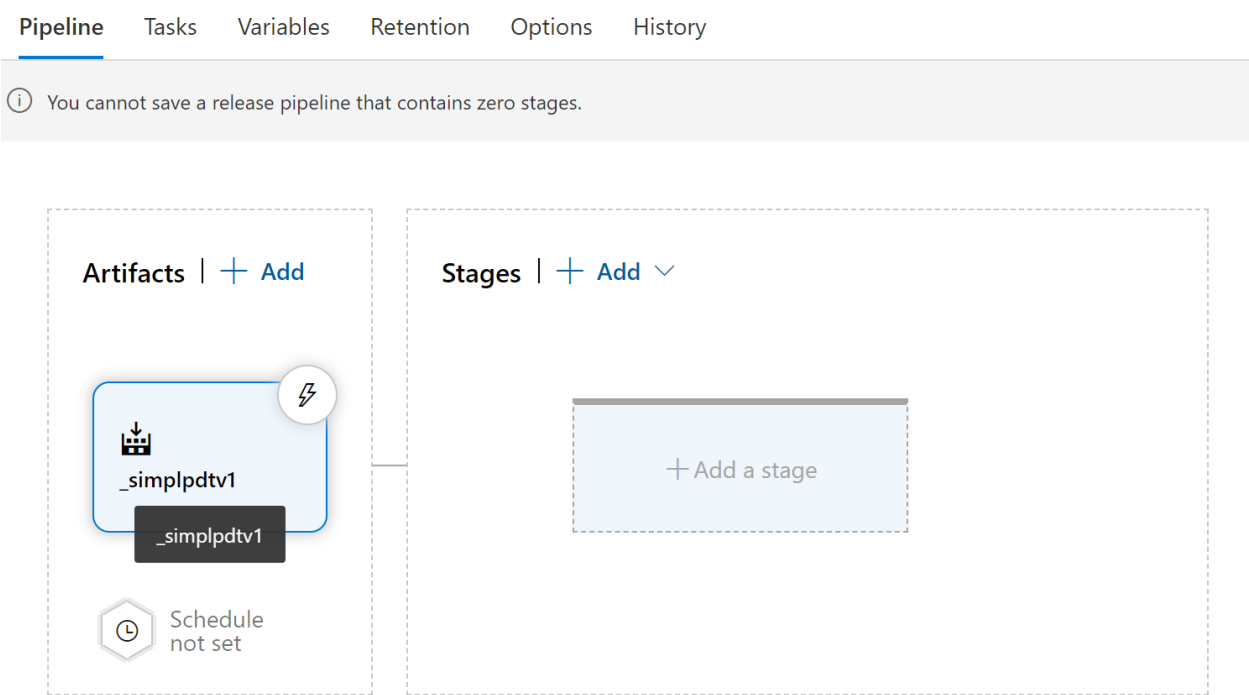
Latest

Source alias *

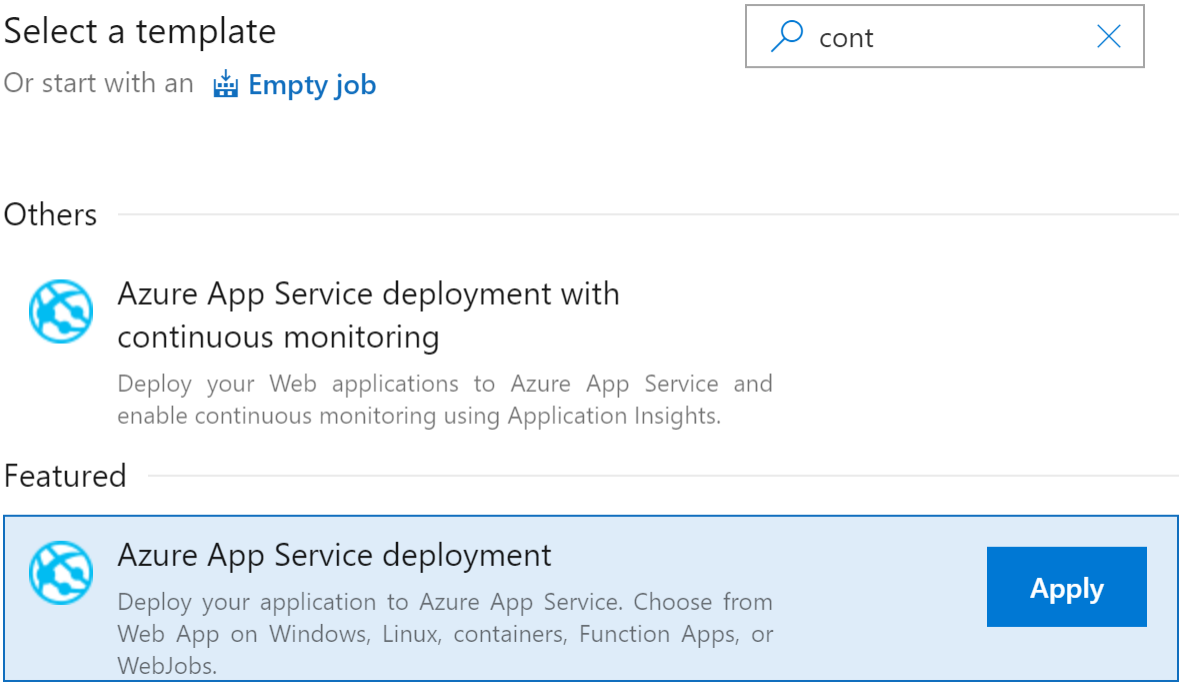
_simplpdtv1

5. Confirm the artifacts selection, by **clicking Add**

6. Your artifact will be completed



7. Next, click Stage / Add a Stage, and select Azure App Service deployment



8. Confirm with Apply

9. Provide the required parameters, based on the resources in your Azure environment

Azure Web App for Containers ⓘ View YAML Remove

Task version 1.*

Display name *

Azure Web App on Container Deploy: contnopwebapp

Azure subscription * ⓘ | Manage

007FFFLearning Labs (0a407898-c077-442d-8e17-71420aa82426)

ⓘ Scoped to subscription '007FFFLearning Labs'

App name * ⓘ

contnopwebapp

☐ Deploy to Slot or App Service Environment ⓘ

Image name ⓘ

nopacr1.azurecr.io/nopcommerce_420_source_nopcommerce_web:$(build.buildnumber)

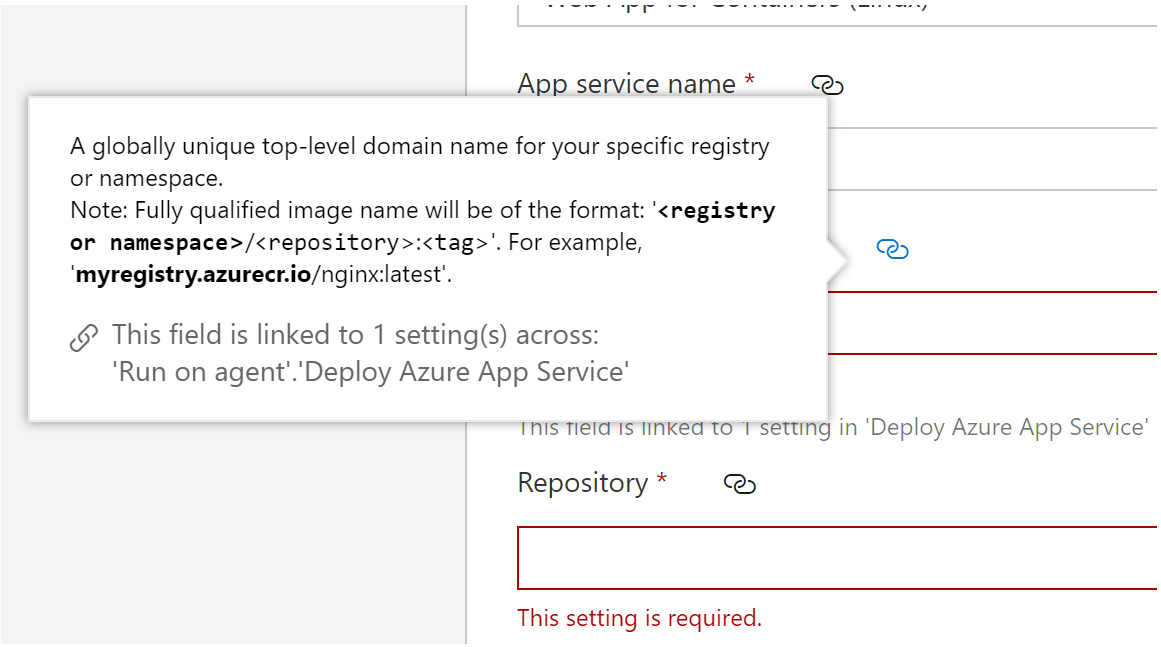
Configuration File ⓘ

Startup command ⓘ

Application and Configuration Settings

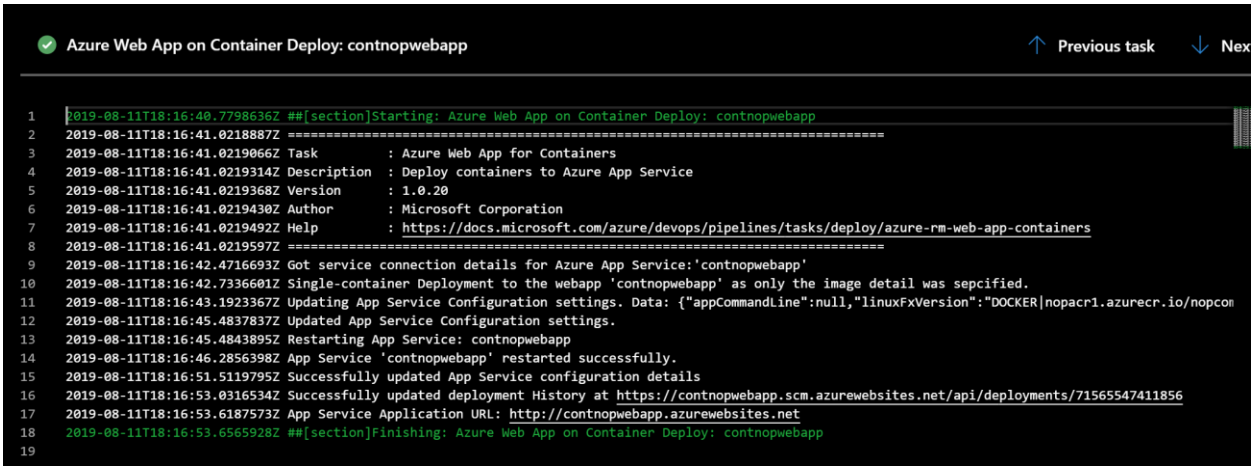
Control Options

10. Note that you have to manually provide the correct string for both Azure Container Registry and Repository you will use

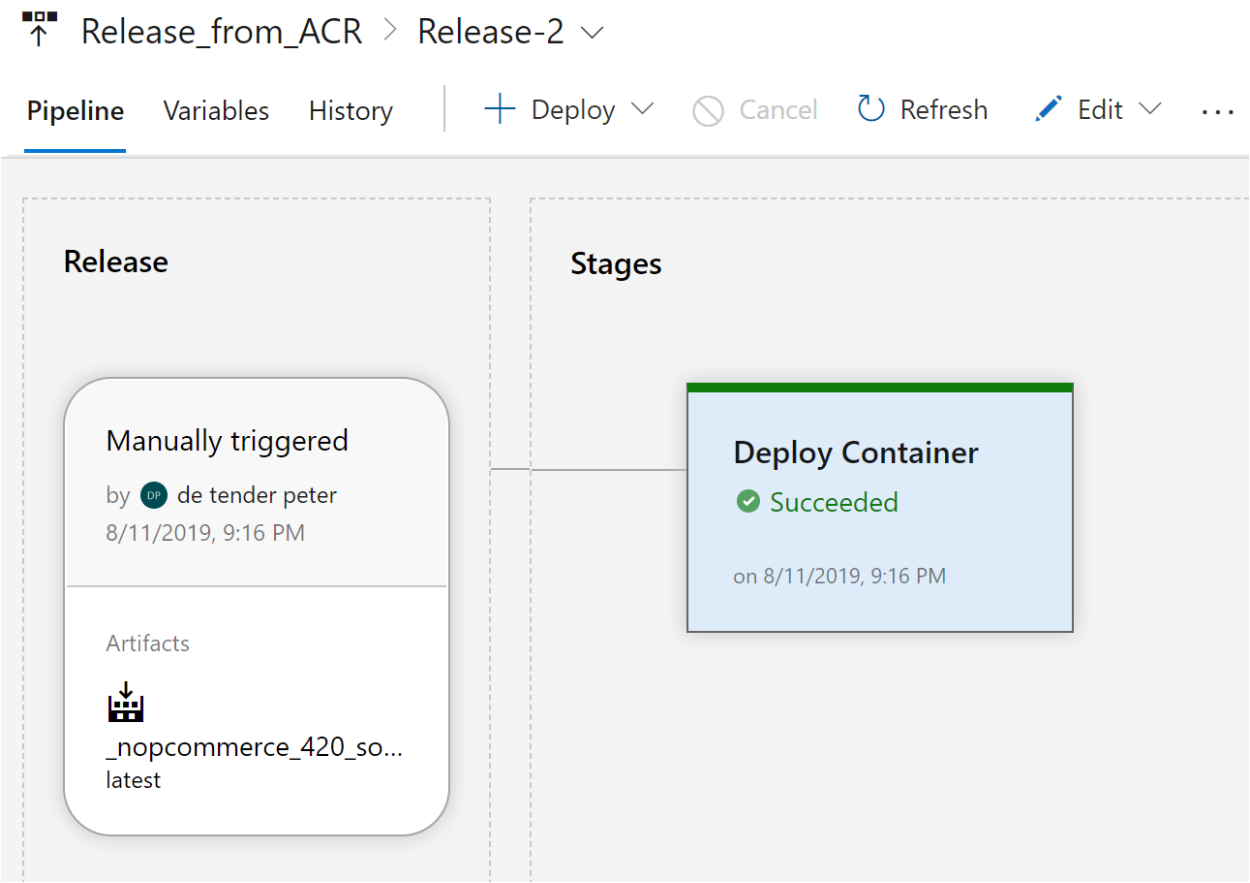


11. Press "Save", followed by "Create Release"

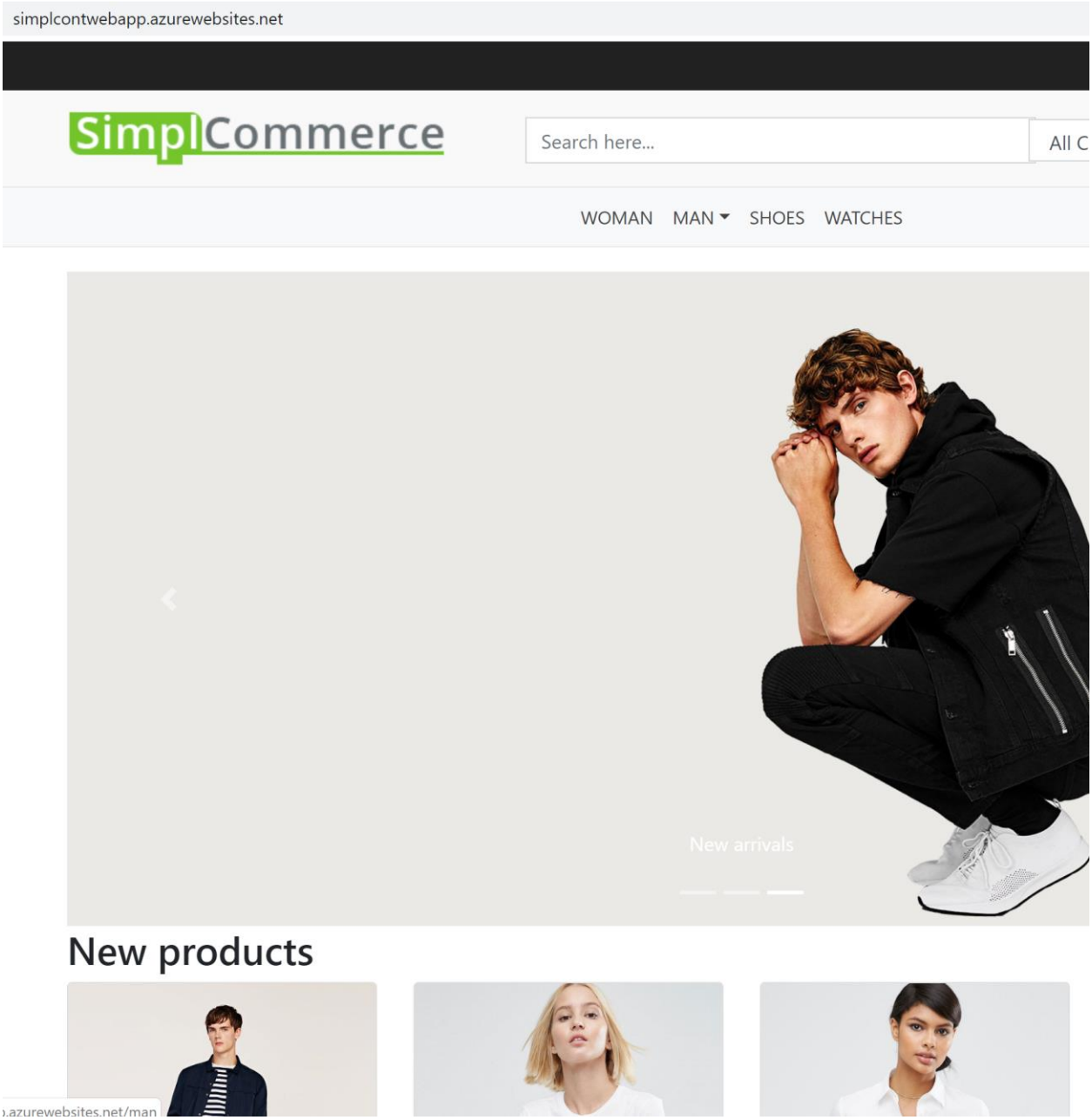
12. This kicks off the release creation; follow the different steps occurring, and wait for them to complete successfully



13. Once the task is complete, you can see its overall status from the Pipeline window



14. Check back in Azure WebApps if your app is running successfully, by connecting to the Azure WebApp URL for this Azure resource

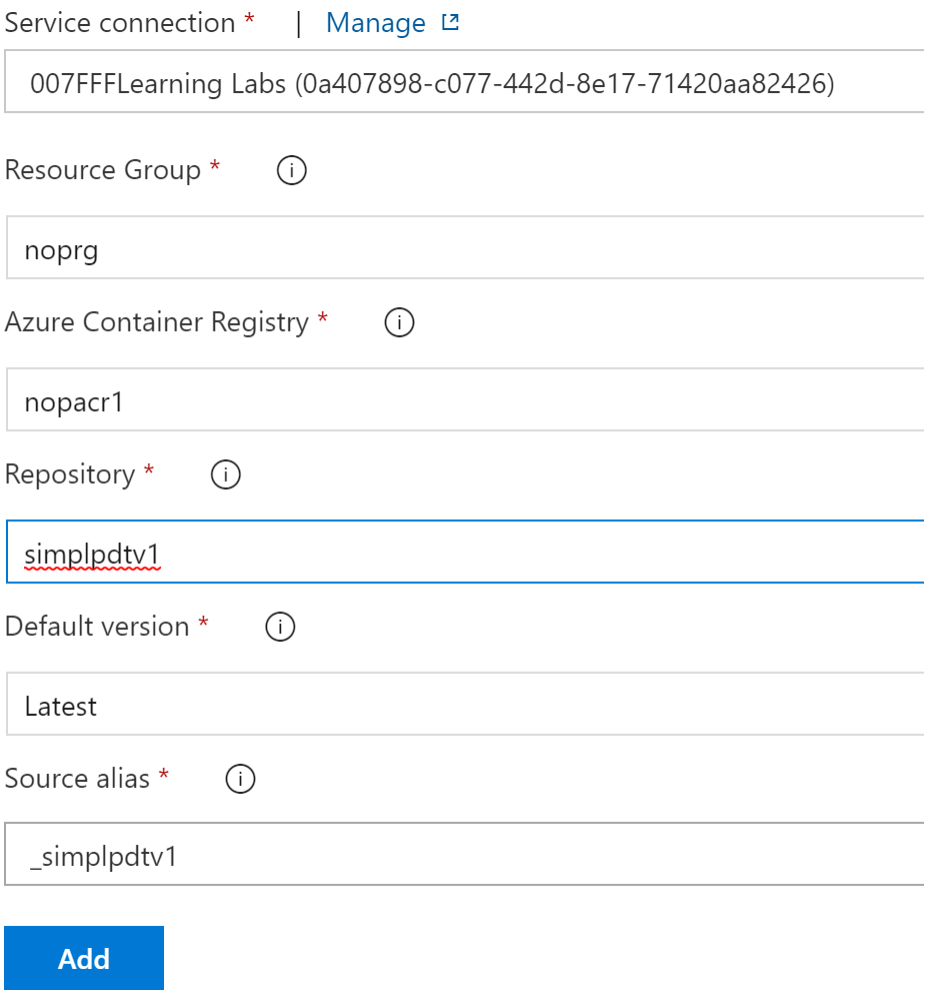


15. This completes the task in which you created a new Azure Pipeline Release, deploying an Azure Web App for Containers, relying on a repository in Azure Container Registry.

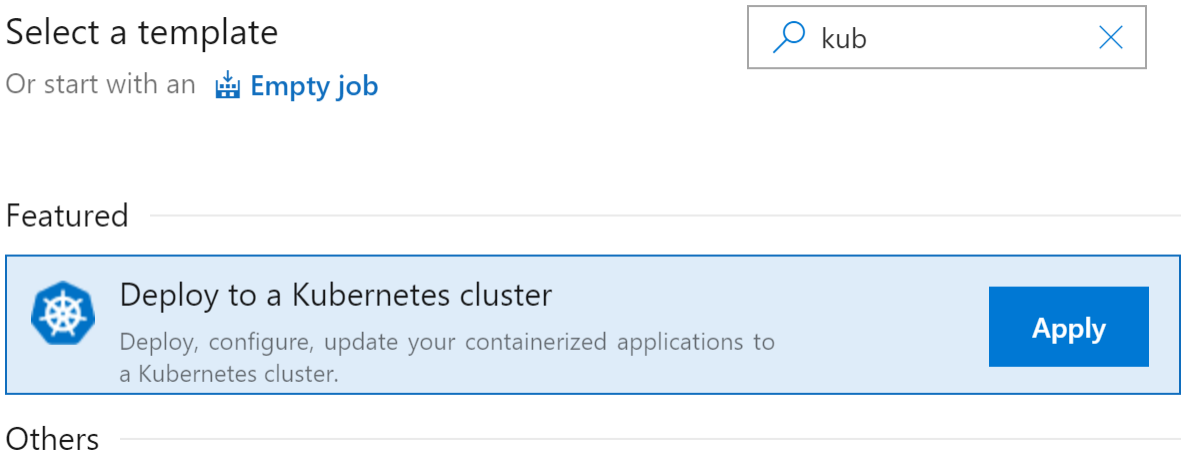
Task 5: Create an Azure DevOps Pipeline to deploy ACR container to Azure Kubernetes Service (AKS)

In this scenario, you will create yet another Azure Release Pipeline, this time pushing a container from ACR into the earlier deployed Azure Kubernetes Service cluster.

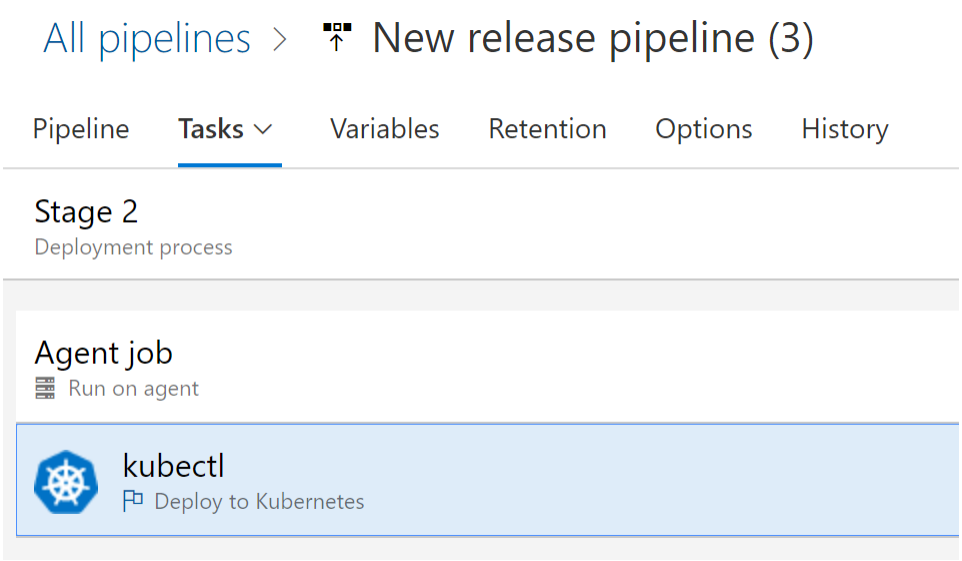
1. From **Azure DevOPs, select Pipeline / Release / New Release Pipeline**
2. Close the appearing **template window**, and return to the **Artifacts**
3. Repeat the steps from the previous task, selecting **Azure Container Registry** as source, and selecting the **ACR registry and container repository you want to use** for this deployment..



4. Next, select "**Add Stage", and** select the **Deploy to a Kubernetes Cluster**" template



5. Confirm by clicking **Apply**



6. **Select "kubectl",** and provide the necessary settings and parameters of the AKS cluster you deployed in a previous lab, knowing you **only need to provide the Kubernetes Service Connection** name

Deploy to Kubernetes ⓘ

View YAML Remove

Task version 0.*

Display name *

deploy_to_aks

Kubernetes service connection ⓘ | Manage

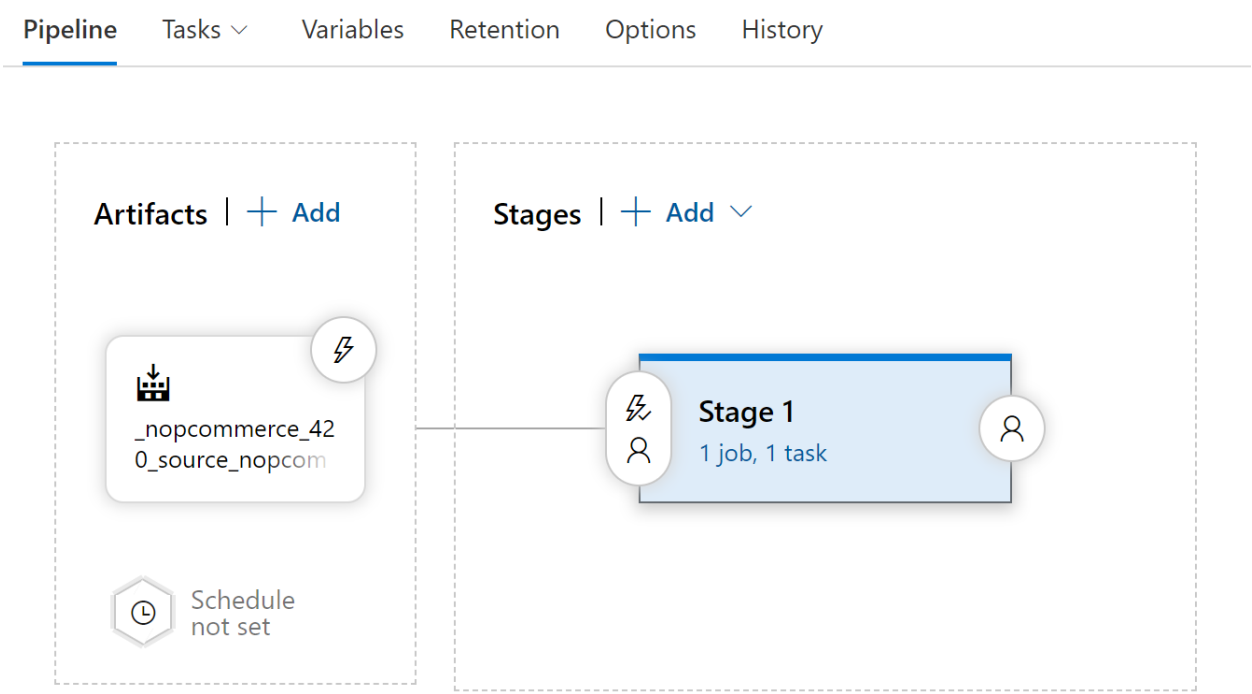
nopaks

New

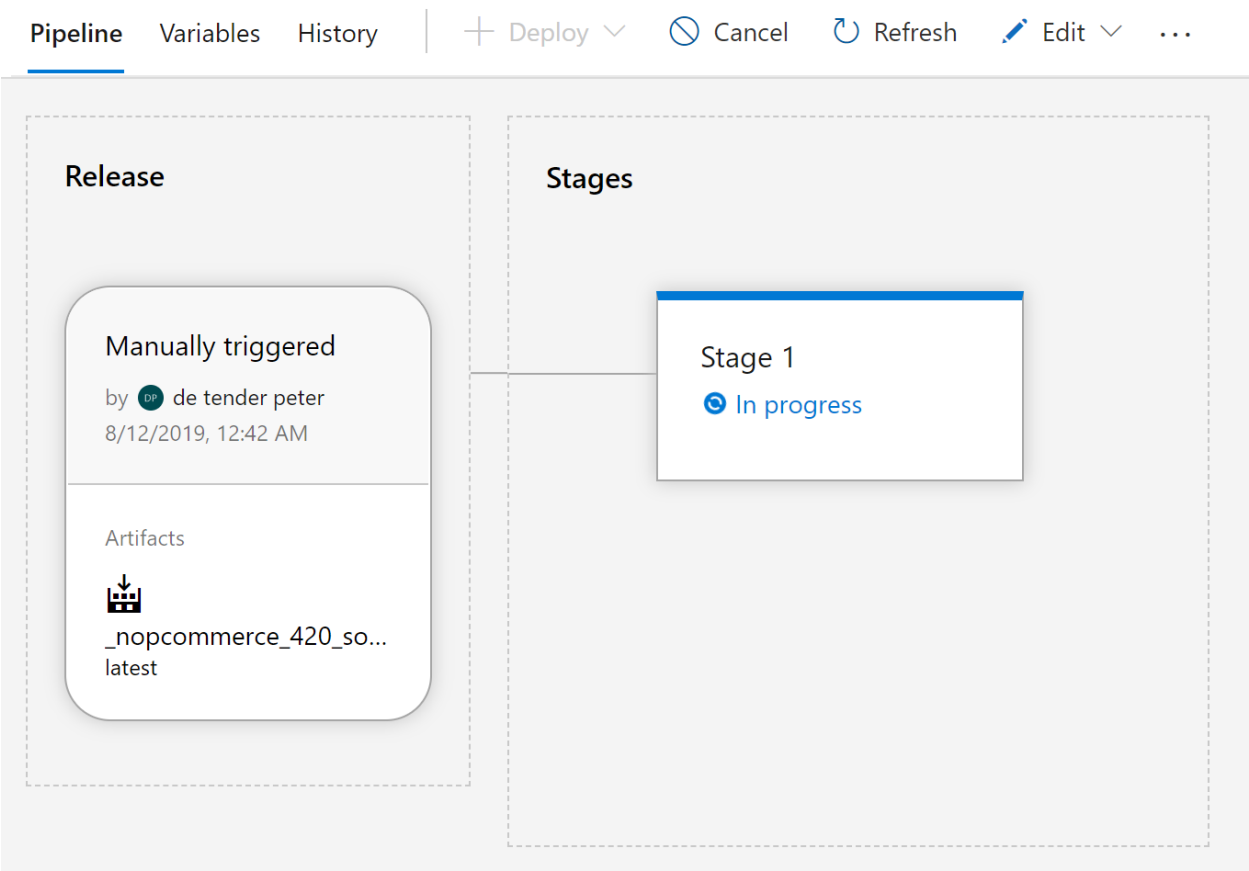
Namespace ⓘ

NOTE: The full deployment process is much much much more powerful and providing many more settings that we will do here, but it is mainly to allow you experiencing what a base deployment Release Pipeline can do, and how to configure it.

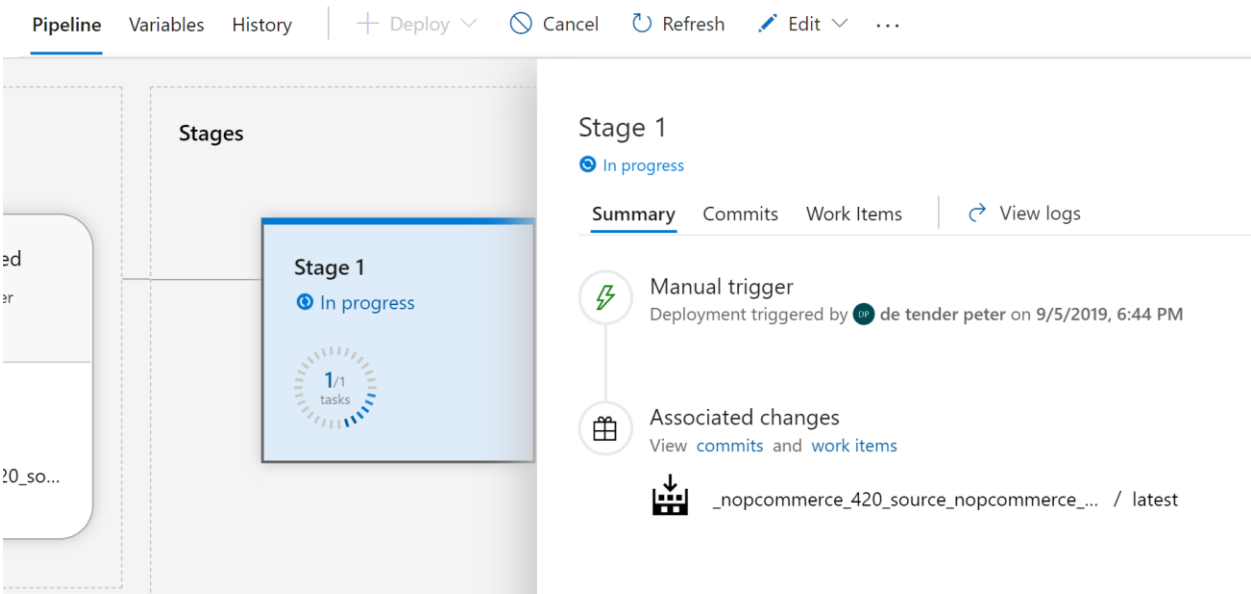
7. **Save** the settings, and validate the Pipeline



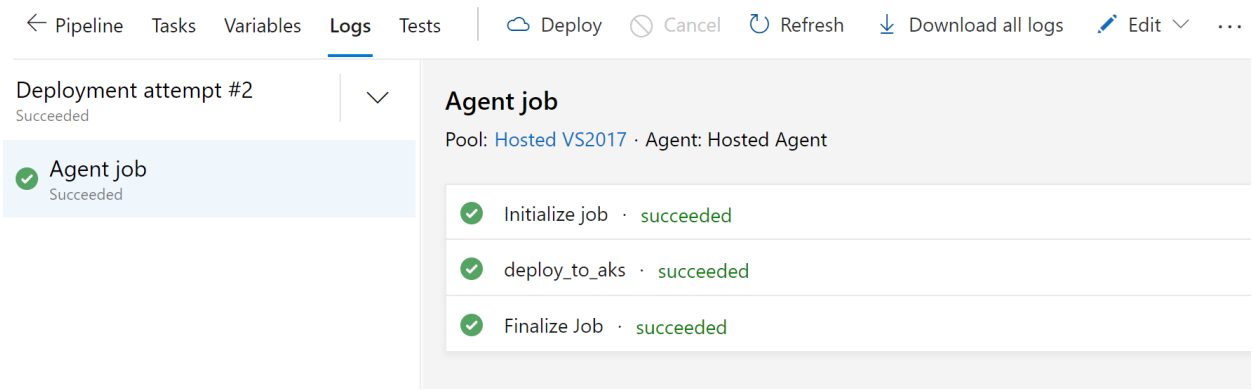
8. Run the Release



9. Seelct "In Progress"



10. Wait for the task to complete successfully



11. This completes the task in which you created a new Azure Release Pipeline for a deployment of an ACR-stored repository, to an existing Azure Kubernetes Cluster.

Summary

In this lab, you performed several tasks around Azure DevOps, starting from the initial creation of an Azure DevOps Organization, followed by creating an Azure DevOps Build Pipeline, using a GitHub repository with an application's source code. In the next task, you created a Release Pipeline, deploying the Build from the previous task, publishing an Azure Web App.

The following tasks involved creating an Azure DevOps Release Pipeline to publish an Azure Container Registry repository image to Azure Container Instance, as well as publishing to Azure Kubernetes Service.

Congrats if you completed all labs with all tasks from all modules. You should know have a real good understanding of Azure and where it can help in your overall digital transformation. Reach out when having any questions, concerns, or want to share overall feedback about the workshop content. Have a nice day!