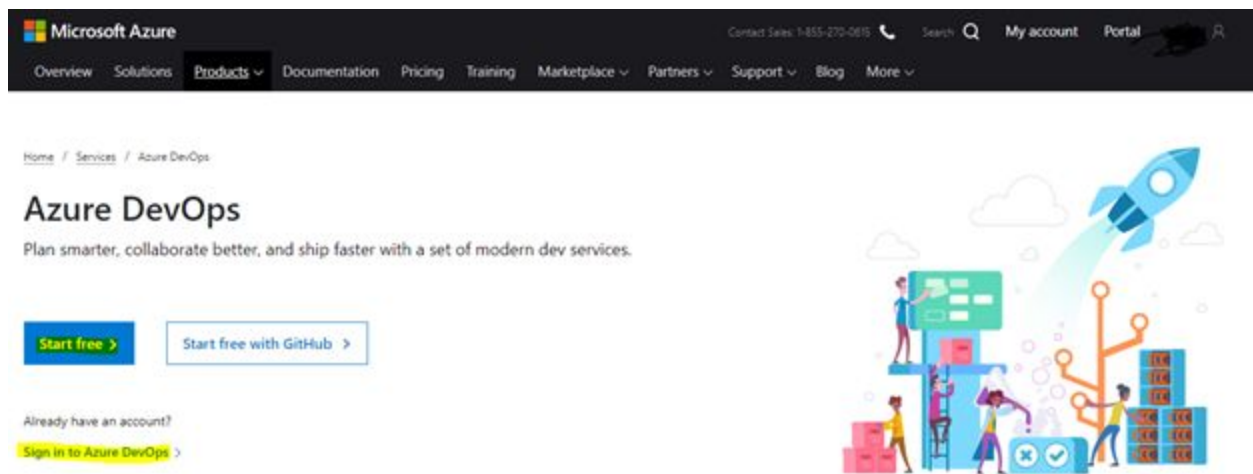


Clear Measure Union DevOps Architecture Azure DevOps Demo Generator Template Implementation

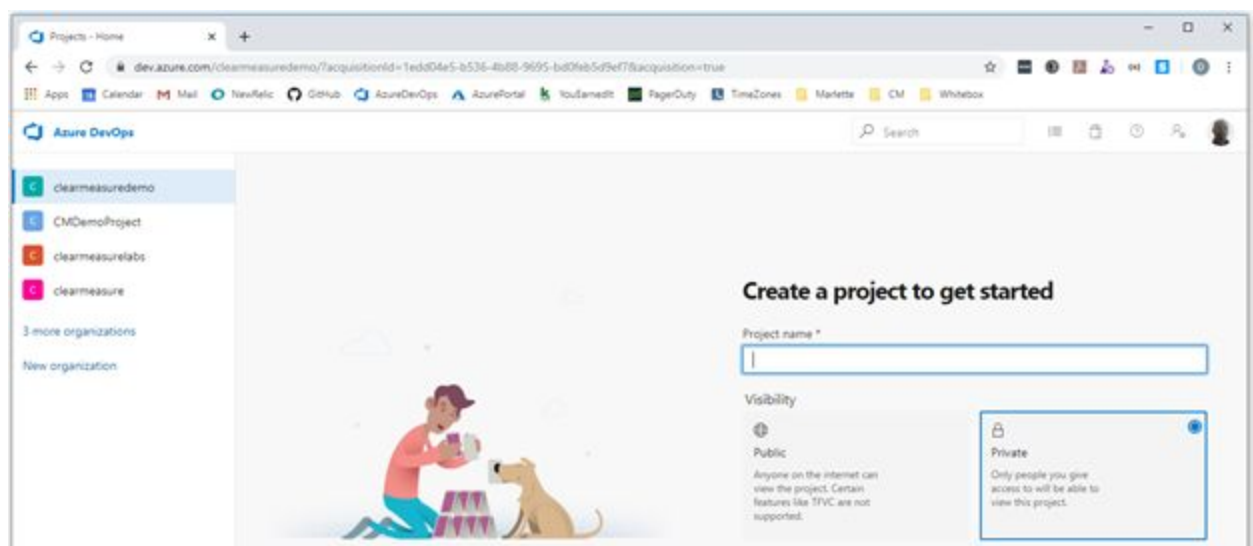
1 - Create new Azure DevOps Organization

<https://azure.microsoft.com/en-us/services/devops/>

Login or register on the Azure DevOps main site.



Follow the prompts to create a new Azure DevOps Organization. Once your new Organization is ready for use, you will be taken to the overview screen. Do not create a new Project at this time.

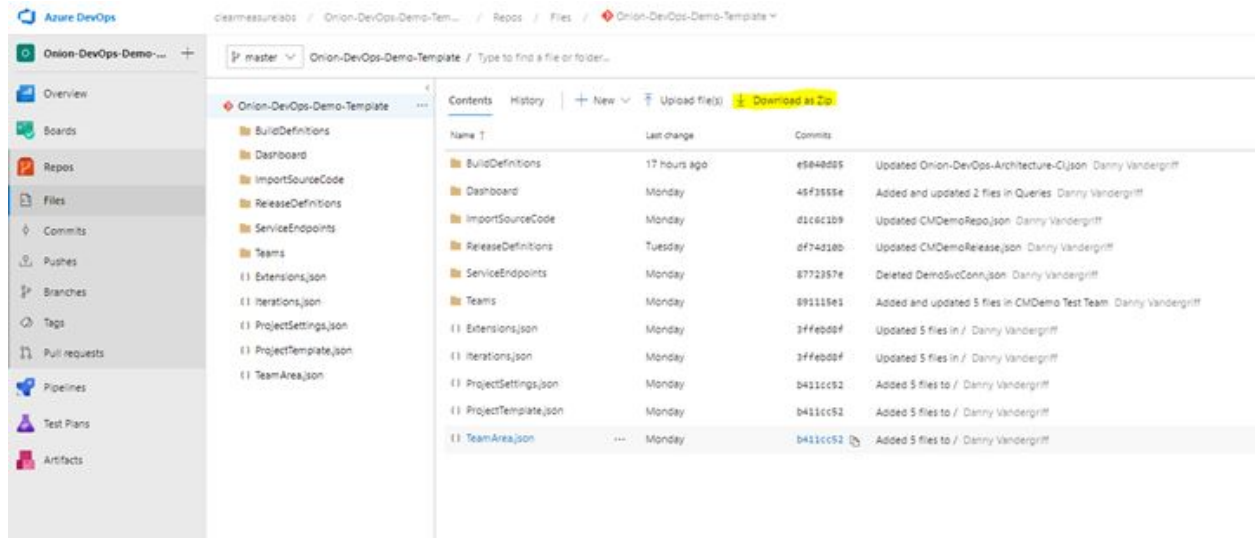


2 - Download the Clear Measure Demo Template package

Navigate to the AzDO repository where the Clear Measure Demo Template package is maintained:

https://dev.azure.com/clearmeasurelabs/Onion-DevOps-dotnet-core2-azure-paas/_git/Onion-DevOps-Template-Dotnet-Core2-Azure-PaaS

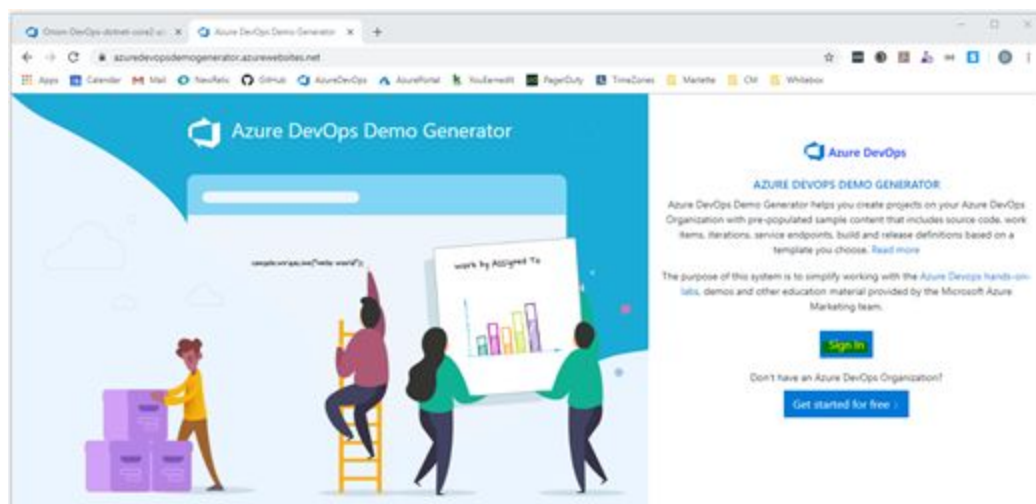
From this screen, click 'Download as Zip' and save the package locally.



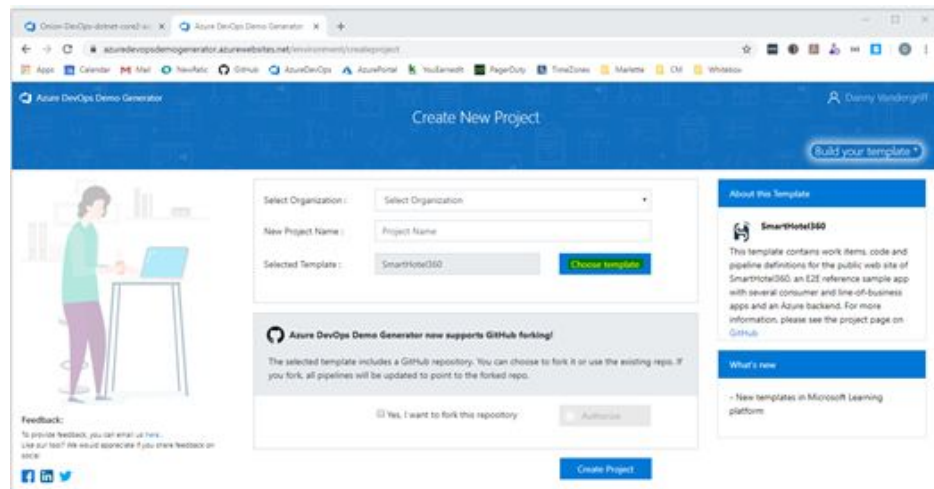
3 - Use Demo Generator Tool to Create New Azure DevOps Project from the Template

Navigate to the new Azure DevOps Demo Generator site, and Sign In:

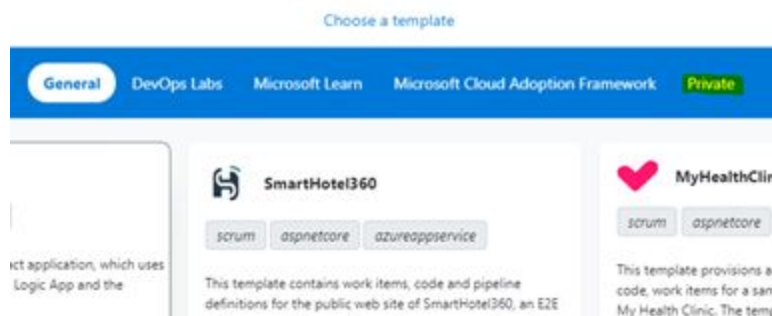
<https://azuredevopsdemogenerator.azurewebsites.net/>



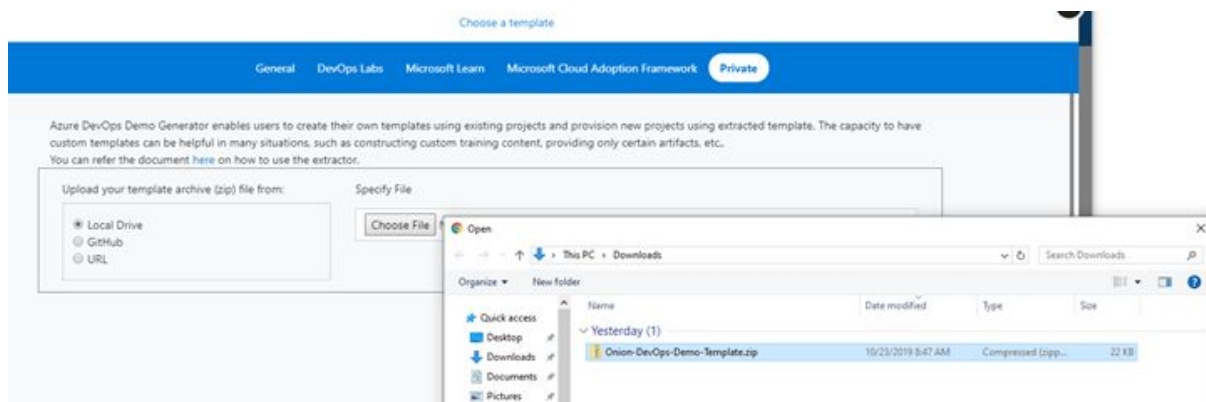
We have found this site to still be somewhat buggy, so it is important to follow the next steps in sequence. First, click the Choose Template button.



There are a few preconfigured templates available for use, but we are going to choose a Private template for the demo.



Click Choose File, then navigate to the Demo template zip file you have saved, and click Submit.



Now you may select your new Azure DevOps Organization from the dropdown menu. Type a name for your new Project, and click Create Project.

Note: Do not put underscores or any special characters in your project name, as this string will ultimately become part of an Azure app service URL.

You will be prompted to add two extensions to your Azure DevOps Organization. The first is a Microsoft extension to add Release Annotations to your Application Insights monitoring. The second is a third-party Build & Release Tools extension that adds powerful step templates for you to add to Pipelines in your Azure DevOps Projects. Check the licensing boxes and Create Project.

Create New Project

Select Organization :

clearmeasuredemo


New Project Name :

newprojectname

Selected Template :

Onion-DevOps-Demo-Template

Choose template

 Azure DevOps Demo Generator now supports GitHub forking!

The selected template includes a GitHub repository. You can choose to fork it or use the existing repo. If you fork, all pipelines will be updated to point to the forked repo.


☐ Yes, I want to fork this repository

Authorize


Verifying if all required extension(s) are installed and enabled

One or more extension(s) is not installed/enabled in your Azure DevOps Organization.

You will need to install and enable them in order to proceed. If you agree with the terms below, the required extensions will be installed automatically for the selected organization when the project is provisioned, otherwise install them manually and try refreshing the page

 Release Annotations for Azure Application Insights - [License Terms](#)

☒ By checking the box I agree, and also on behalf of all users in the organization, that our use of the extension(s) is governed by the [Microsoft Online Services Terms](#) and [Microsoft Online Services Privacy Statement](#)

 Build & Release Tools - [License Terms](#)

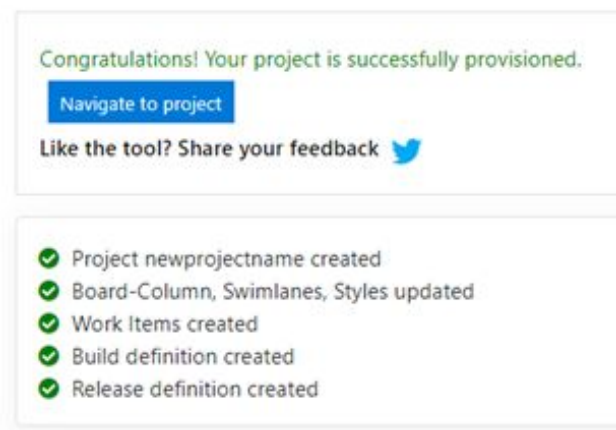
☒ The extension(s) are offered to you for your use by a third party, not Microsoft. The extension(s) is licensed separately according to its corresponding License Terms. By continuing and installing those extensions, you also agree to those License Terms.

Create Project

A status indicator will appear as the template is imported.

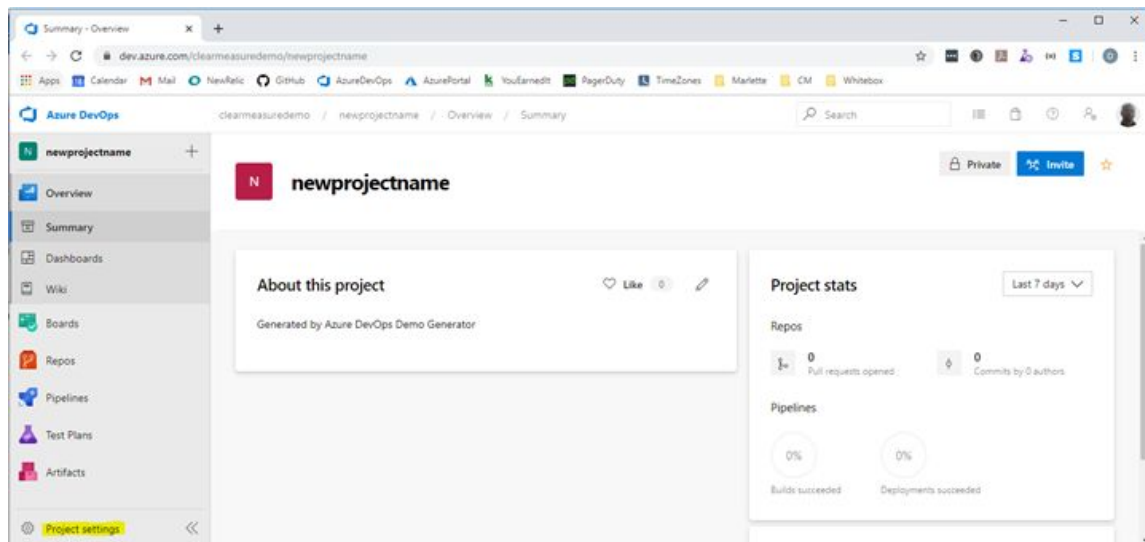


When it is complete, you will get a link to Navigate to Project. Click it and we will begin configuring your new Project.

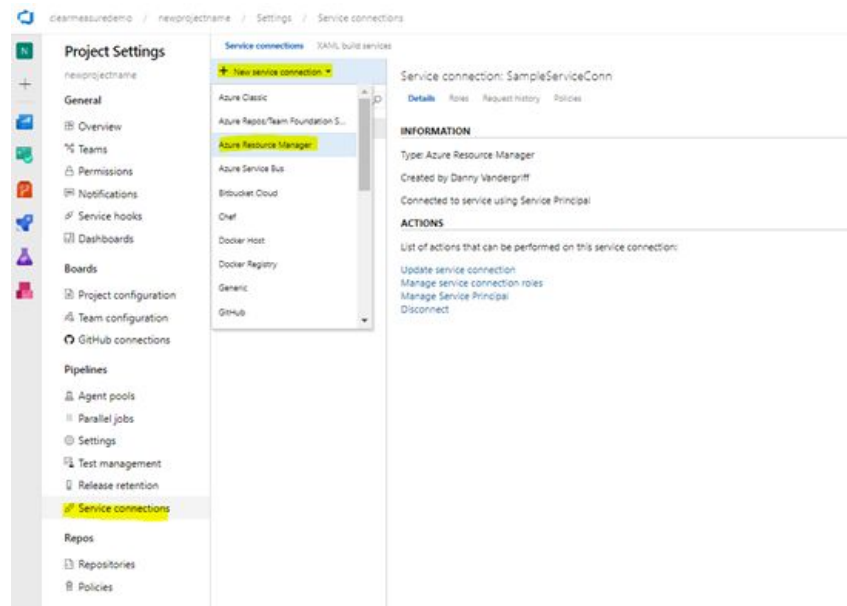


4 - Configure Project Level Settings

We will first need to create an Azure Resource Manager Service Connection for your Project that will allow your Pipelines to create resources in Azure. Navigate to Project Settings.



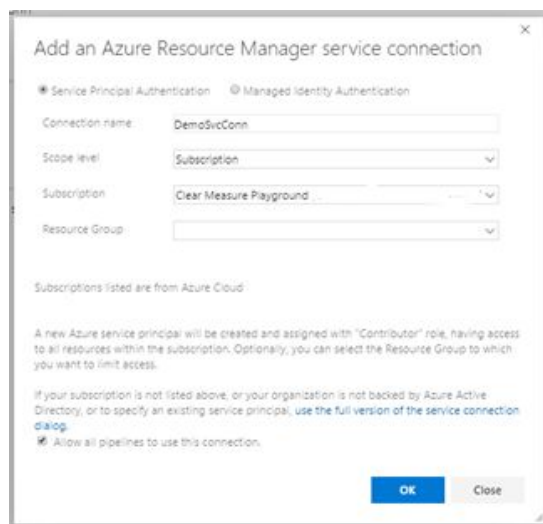
Then select Service Connections from the menu / add a New Service Connection / Azure Resource Manager. Note that there is an unconfigured SampleServiceConn, because the Demo Generator tool requires a Service Connection for the import to function. You can disregard it.



Give your new Service Connection this name: **AzureSvcConn**

Predefined Build and Release Pipeline variables are configured to use this name.

Scope level should remain at **Subscription**. Use the dropdown to locate an Azure Subscription where you have enough privileges in the Subscription and Azure Active Directory to add a Service Principal. Leave Resource Group blank, ensure the check for 'Allow all pipelines to use this connection' is checked, and click OK.

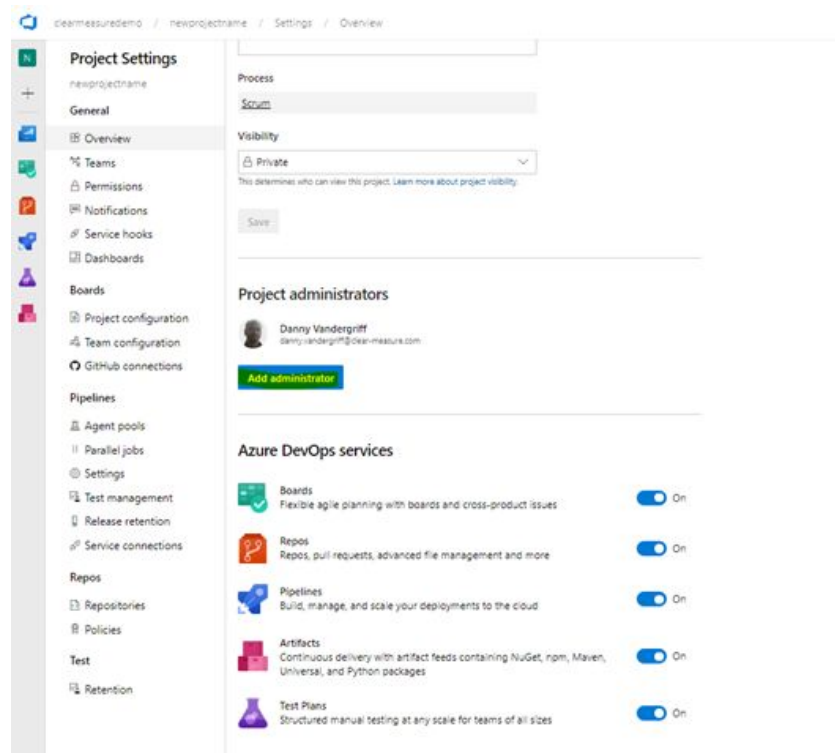


Note: If you get an error about inadequate permissions to create a Service Principal in your Azure subscription, work with an Azure administrator in your organization to create one manually. You will need subscription details, Azure Active Directory tenant details, and the ID and Key for the Service Principal to proceed. Directions for creating a Service Principal can be found here:

<https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-service-principal-portal>

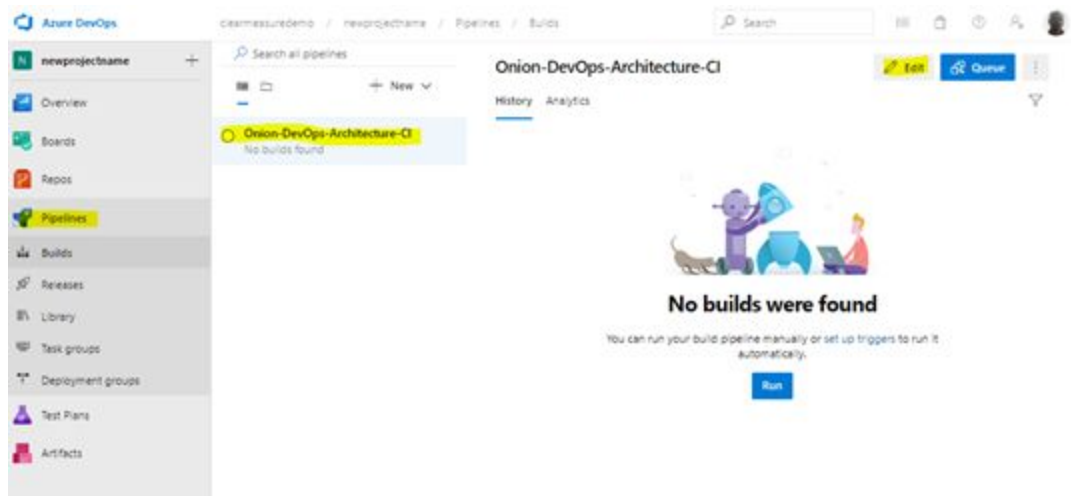
Another Project Setting that you may want to consider is adding additional Project Administrators to your Organization under the Overview menu.

Verify that all Azure DevOps Services are set to 'On' for the demo.

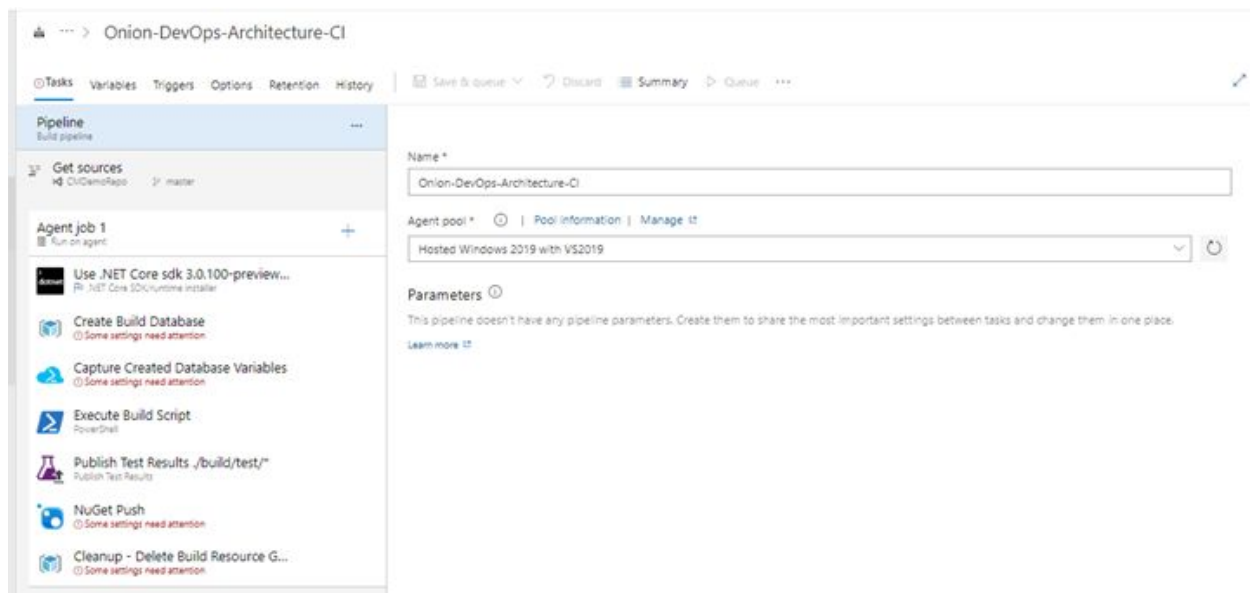


5 - Configure Onion-DevOps-Architecture CI Build Pipeline and Run Initial Continuous Integration Build

We will now begin setting up the CI Build Pipeline for the initial build, test, and package of the Onion DevOps Architecture application. Browse to Pipelines / Builds / and Edit the Onion-DevOps-Architecture-CI pipeline.



You will see the job steps listed, with four of them indicating 'Some settings need attention.'



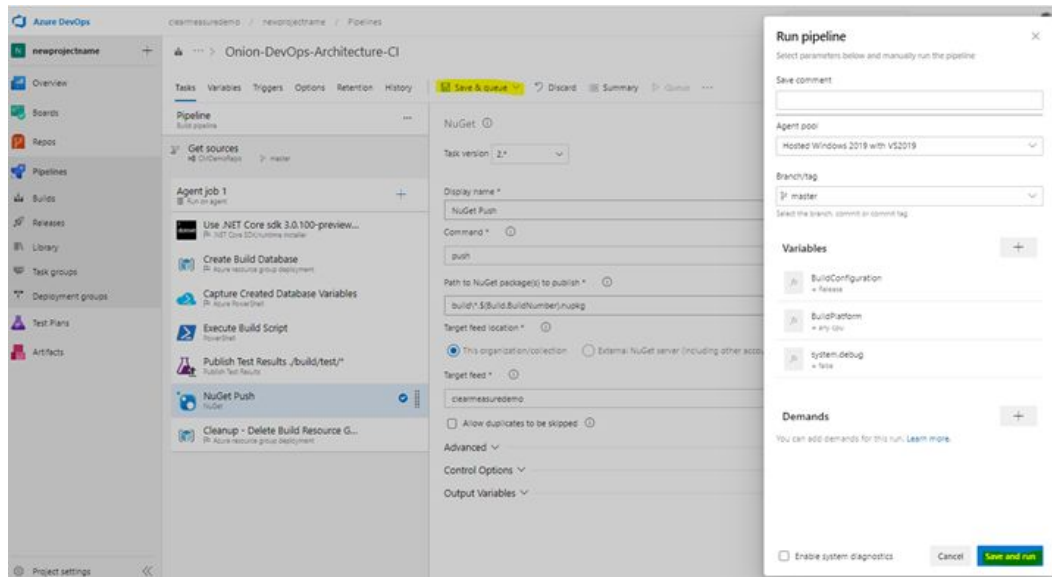
Click each step requiring attention. For the steps that require the 'Azure Subscription' to be populated, use the dropdown to locate the AzureSvcConn Service Connection that you created earlier.

The screenshot shows the configuration for the 'Azure resource group deployment' task. The left sidebar lists the pipeline steps: 'Get sources', 'Agent job 1', 'Use .NET Core sdk 3.0.100-preview...', 'Create Build Database', 'Capture Created Database Variables', 'Execute Build Script', 'Publish Test Results ./build/test/*', 'NuGet Push', and 'Cleanup - Delete Build Resource G...'. The 'Create Build Database' step is highlighted with a yellow background and a blue checkmark. The main configuration area for the 'Azure resource group deployment' task shows the 'Task version' as 2.0. The 'Display name' is 'Create Build Database'. Under 'Azure Details', the 'Azure subscription' dropdown is open, showing a list of available Azure service connections: 'SampleServiceConn' and 'DemoSvcConn'. Below this, a list of available Azure subscriptions is shown, including 'Microsoft Azure Sponsorship 1', 'DV - Visual Studio Enterprise - MPN', 'MS Azure 2019 Sponsorship', and 'Microsoft Azure Sponsorship'. The 'DemoSvcConn' service connection is highlighted in yellow.

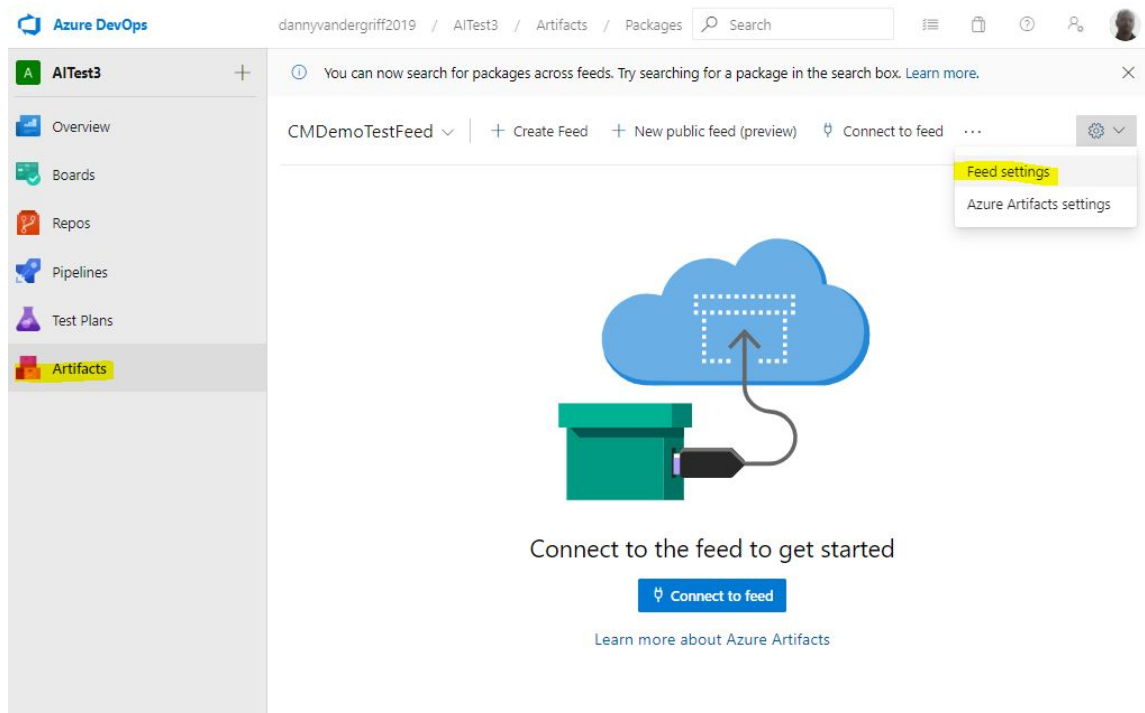
The NuGet Push step will need to have a Target Feed defined. The default Feed is created with the AzDO Organization, and shares the name. Select it for the Demo.

The screenshot shows the configuration for the 'NuGet Push' task. The left sidebar lists the pipeline steps: 'Get sources', 'Agent job 1', 'Use .NET Core sdk 3.0.100-preview...', 'Create Build Database', 'Capture Created Database Variables', 'Execute Build Script', 'Publish Test Results ./build/test/*', 'NuGet Push', and 'Cleanup - Delete Build Resource G...'. The 'NuGet Push' step is highlighted with a yellow background and a blue checkmark. The main configuration area for the 'NuGet Push' task shows the 'Task version' as 2.0. The 'Display name' is 'NuGet Push'. The 'Command' is 'push'. The 'Path to NuGet package(s) to publish' is 'build*. \$(Build.BuildNumber).nupkg'. The 'Target feed location' is set to 'This organization/collection'. The 'Target feed' dropdown is open, showing a list of available target feeds: 'ClearmeasureDemo' and 'ClearmeasureDemo'. The 'ClearmeasureDemo' target feed is highlighted in yellow. The 'Allow duplicates to be skipped' checkbox is checked. The 'Advanced' section is expanded, showing 'Control Options' and 'Output Variables'.

Then click the 'Save & queue' button, then 'Save and run' to initiate the first CI Build.



Immediately, you will need to return to Artifacts to reconfigure the Feed for your project. Azure DevOps has a reported bug where an option to allow builds to push packages into feeds is not available - until the build has been triggered! First navigate to Artifacts / Feed Settings.



Select Permissions.

The screenshot shows the Azure DevOps interface for the 'CMDemoTestFeed' feed settings. The left sidebar contains a navigation menu with 'Artifacts' selected. The main content area shows the 'Permissions' tab, which is currently empty. Below the feed details, there is a section for 'Deleted packages' with a checkbox labeled 'Hide deleted package versions' which is checked.

Feed details | **Permissions** | Views | Upstream sources | Delete feed

Name: CMDemoTestFeed

Description:

Deleted packages

Because feeds are immutable, deleted package versions are shown by default as a reminder that the version number is taken and can't be published again. You can hide these versions, but the version numbers will continue to be reserved. When selected, deleted packages will be hidden from Administrators. Contributors and Readers never see deleted packages.

☒ Hide deleted package versions

Click the ellipsis at the end of the navigation bar, and select “Allow project-scoped builds” if it is not already selected.

The screenshot shows the Azure DevOps interface for the 'CMDemoTestFeed' feed settings. The left sidebar contains a navigation menu with 'Artifacts' selected. The main content area shows the 'Permissions' tab, which is populated with a table of users and groups. A dropdown menu is open at the end of the navigation bar, showing the option 'Allow project-scoped builds' which is highlighted.

Feed details | **Permissions** | Views | Upstream sources | + Add users/groups | Delete | ...

User/Group	Permission	Inherited
Danny Vandergriff	Allow builds and releases	
[dannyvandergriff2019]\Project Collection Administrators	Owner	✓
[AITest3]\Project Administrators	Owner	
Project Collection Build Service (dannyvandergriff2019)	Contributor	
AITest3 Build Service (dannyvandergriff2019)	Contributor	
[AITest3]\Contributors	Contributor	

Return to the Pipeline to monitor the status of your Build. If the build has failed at the “NuGet Push” step, it is likely due to the missing permission noted above. Address the issue with the Feed, and Queue a new Build.

1.0.1.0 Onion-DevOps-Architect: x +

dev.azure.com/clearmeasuredemo/newprojectname/_build/results?buildId=1

newprojectname

Overview

Boards

Repos

Pipelines

Buils

Releases

Library

Task groups

Deployment groups

Test Plans

Artifacts

Project settings

Logs Summary Tests

Agent job 1
Pool: Azure Pipelines - Agent: Hosted Agent
Started: 10/24/2019, 12:49:31 AM
2m 51s

Initialize job - succeeded 7s

Checkout - succeeded 8s

Use .NET Core sdk 3.0.100-preview3-010431 - succeeded 15s

Create Build Database 2m 19s

Task : Azure resource group deployment
Description : Deploy an Azure Resource Manager (ARM) template to a resource group and manage virtual machines
Version : 2.157.4
Author : Microsoft Corporation
Help : https://docs.microsoft.com/azure/devops/pipelines/tasks/deploy/azure-resource-group-deployment
Checking if the following resource group exists: Build00--Onion-DevOps-Architecture-CI-1.0.1.0.
Resource group exists: false.
Creating Resource group: Build00--Onion-DevOps-Architecture-CI-1.0.1.0.
Resource group created successfully.
Creating deployment parameters.
The detected encoding for file 'D:\a\1\src\Database\databaseAMM.json' is 'utf-8'.
Starting deployment.
Deployment name is databaseAMM-20191024-05005-33dc

Capture Created Database Variables - pending

Execute Build Script - pending

Publish Test Results /build/test/ - pending

NuGet Push - pending

Cleanup - Delete Build Resource Group - pending

All steps must be complete and showing “all green” before proceeding.

Note: the CI Build must be complete before any additional configuration of the Release pipeline steps can begin! This normally takes around 5-10 minutes.

Azure DevOps

clearmeasuredemo / newprojectname / Pipelines / Buils / Onion-DevOps-Architecture... / #1.0.1.0

newprojectname

Overview

Boards

Repos

Pipelines

Buils

Releases

Library

Task groups

Deployment groups

Test Plans

Artifacts

Project settings

#1.0.1.0: Updated chromedriver.exe

Manually run today at 12:49 am by Danny Vandergriff @ CVDemoRepo [master @ acd5618]

Release

All logs

Logs Summary Tests

Agent job 1
Pool: Azure Pipelines - Agent: Hosted Agent
Started: 10/24/2019, 12:49:31 AM
9m 17s

Prepare job - succeeded +1s

Initialize job - succeeded 7s

Checkout - succeeded 8s

Use .NET Core sdk 3.0.100-preview3-010431 - succeeded 15s

Create Build Database - succeeded 2m 22s

Capture Created Database Variables - succeeded 26s

Execute Build Script - succeeded 2m 29s

Publish Test Results /build/test/ - succeeded 9s

NuGet Push - succeeded 25s

Cleanup - Delete Build Resource Group - succeeded 2m 51s

Post-job: Checkout - succeeded +1s

Finalize job - succeeded +1s

Report build status - succeeded +1s

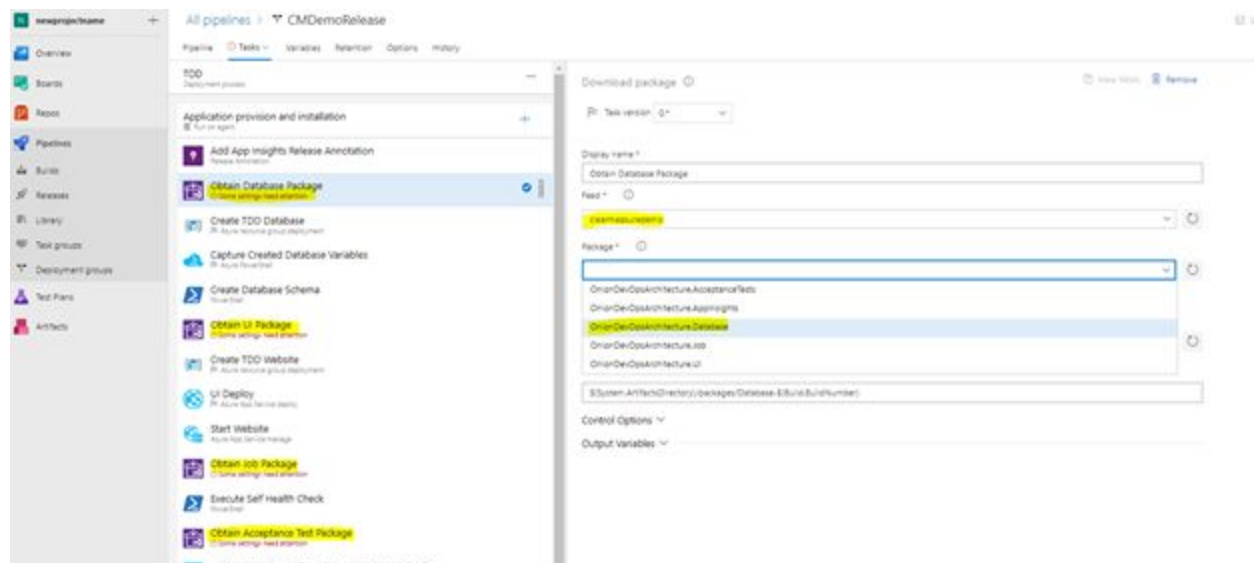
6 - Configure the CMDemoRelease Pipeline and deploy the Onion DevOps Architecture Demo application

Browse to Pipelines / Releases / and Edit the CMDemoRelease pipeline.



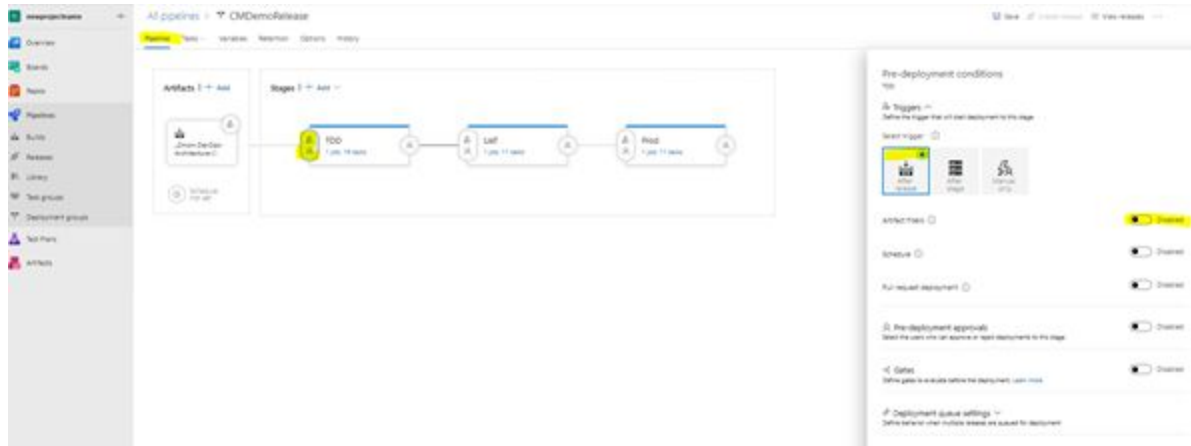
Navigate to the Tasks tab, select TDD, and note that the 'Obtain Package' steps all require attention. Each one of them will need to be updated with a Feed name and the Package name.

Note: The name of the Release step will indicate which package you need to choose from the Feed. Choose them carefully, or your Release will fail on deployment.



Update the Feed and Package for every Release steps that need attention in the TDD, UAT, and Prod Stages.

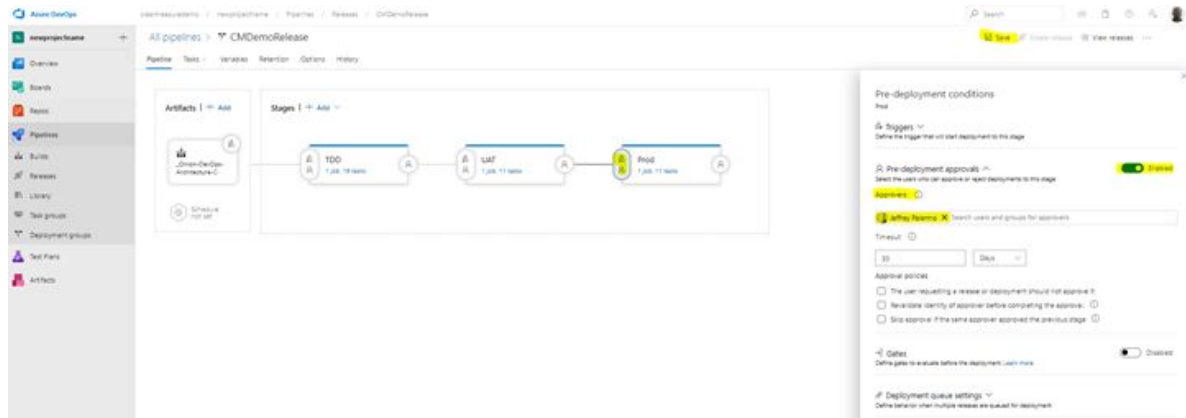
We will make the TDD Stage deploy automatically any time a CI Build creates a successful Release. Navigate to the Pipeline tab, and click the icon on the left side of the TDD Stage. Add a new Trigger for 'After Release'. We will not put any Artifact Filters on this trigger, so this configuration will automatically trigger a deployment to TDD every time a new package is available.



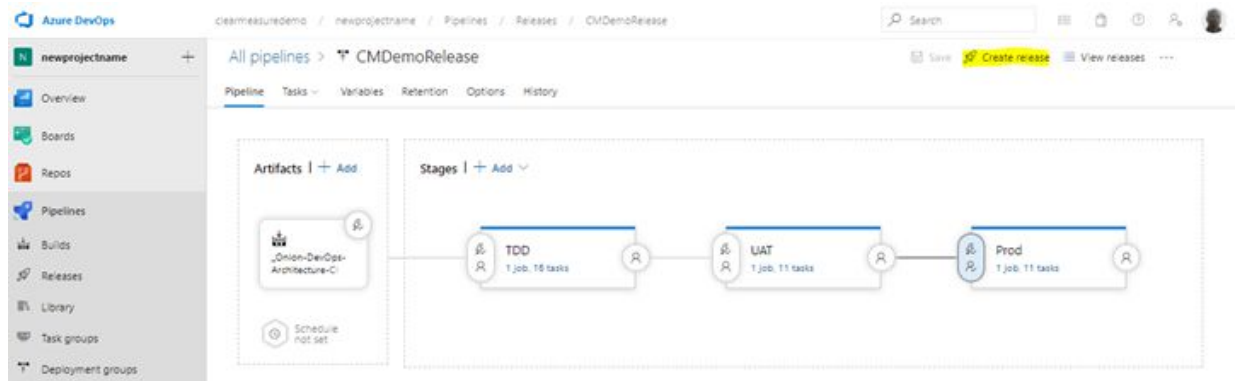
The UAT Stage has already been configured to deploy automatically after a successful TDD deployment, and it also has an Artifact Filter enabled so that only build artifacts derived from the Master branch can be released to UAT.



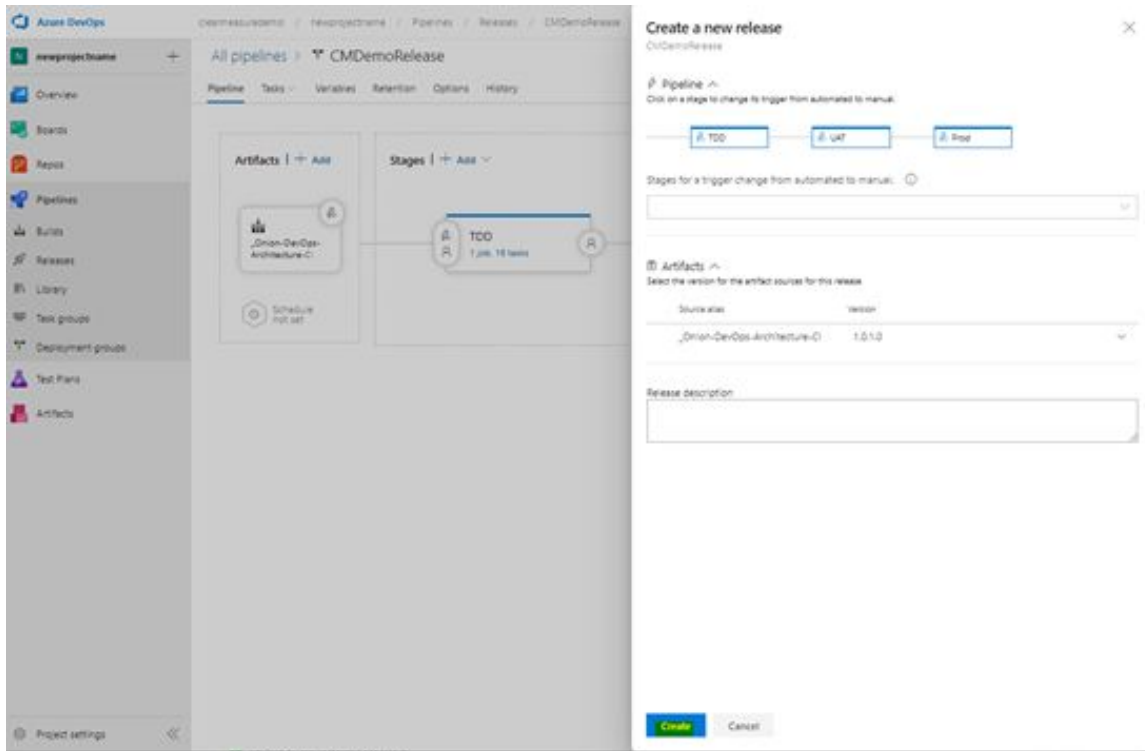
Finally, we will add a pre-deployment Approval for Production. Click the icon on the left side of the Prod Stage. Enable a pre-deployment Approval, and add one or more Approvers from your AzDO Organization. Then click Save.



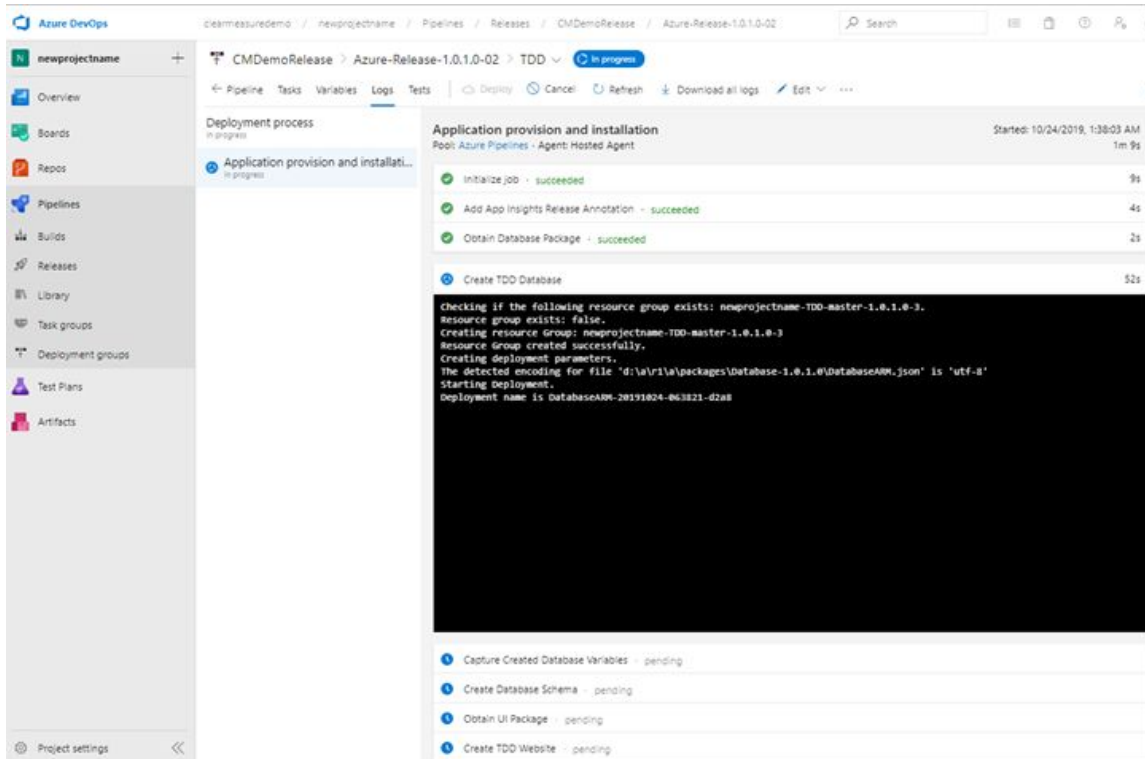
Manually create a Release to trigger a deployment to TDD.



You will be prompted to verify you want to Create the Release.



You can watch the deployment in real-time by browsing to the Log tab for the Release.



Each successive Stage is triggered by successful completion of the prior Stage, so the Release pipeline should be fully automated (with a manual approval for release to Production).

