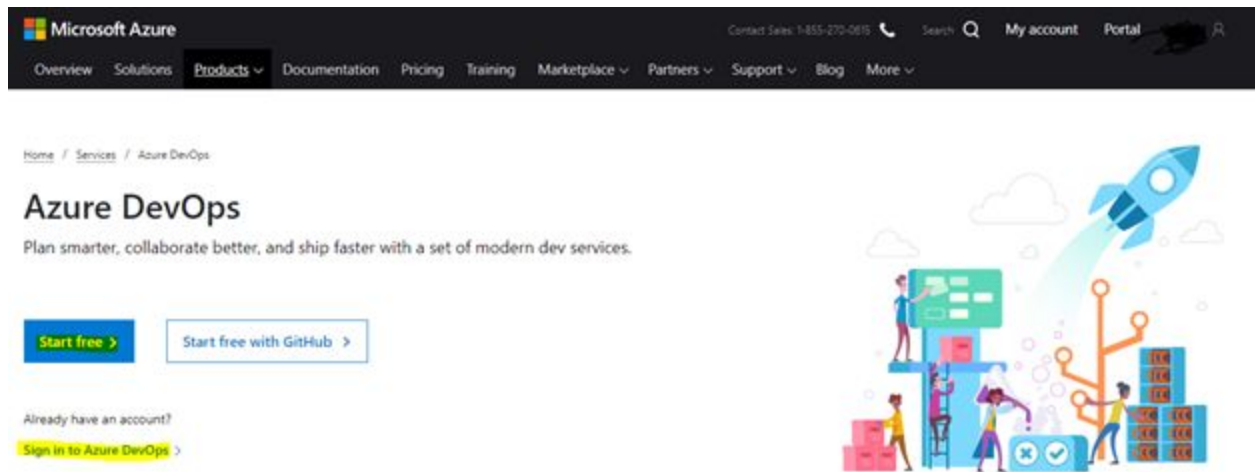


Clear Measure Union DevOps Architecture Azure DevOps Demo Generator Template Implementation

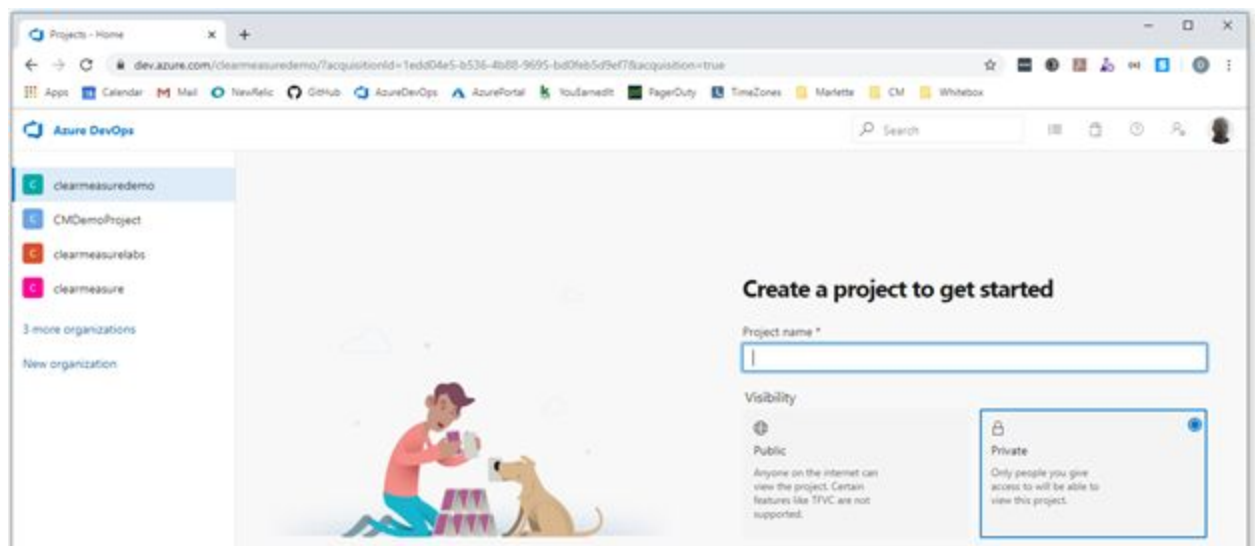
1 - Create new Azure DevOps Organization

<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.

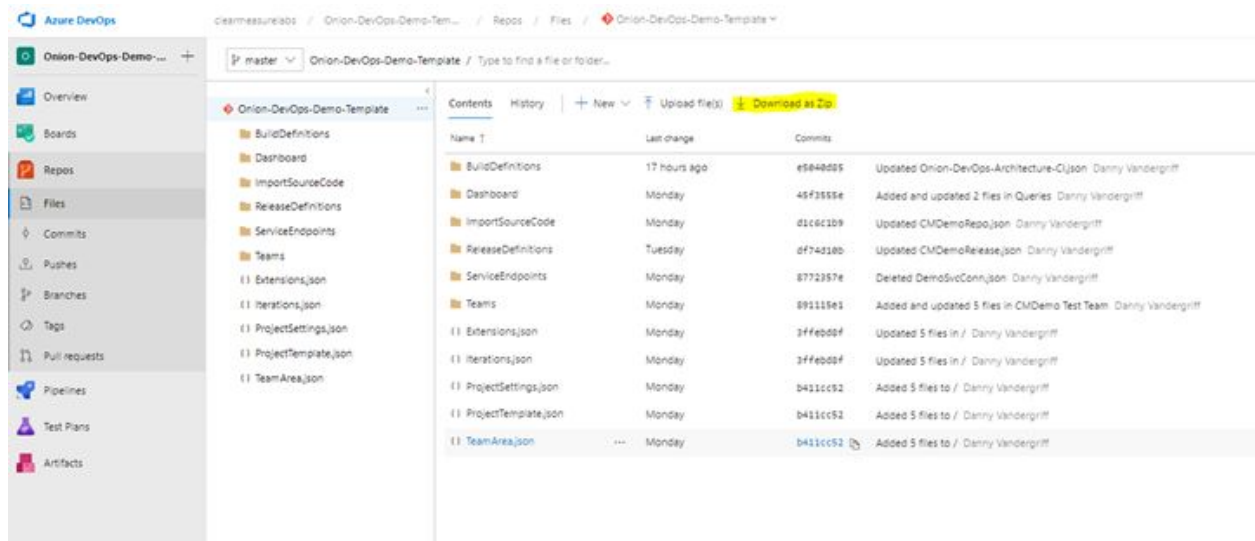


2 - Download the Clear Measure Demo Template package

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

https://dev.azure.com/clearmeasurelabs/_git/Onion-DevOps-Demo-Template

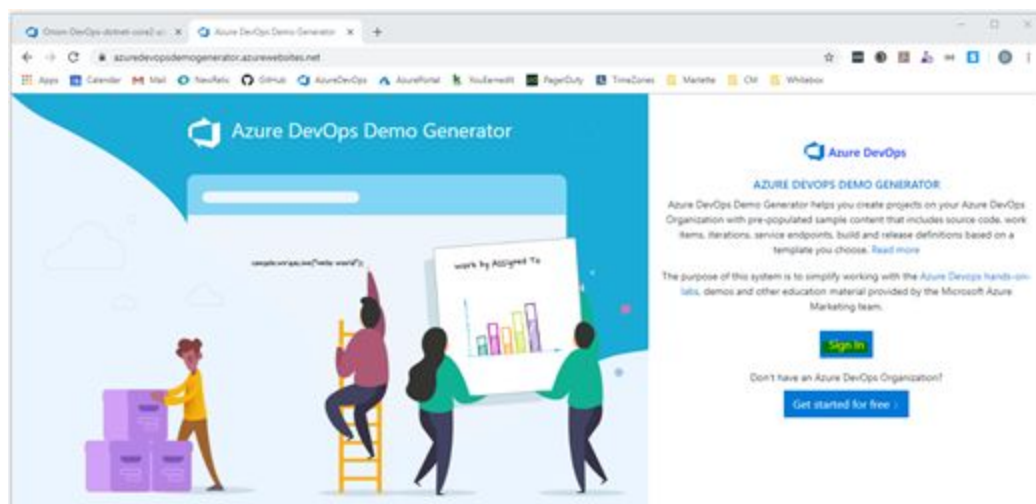
From this screen, click 'Download as Zip' and save the package locally.



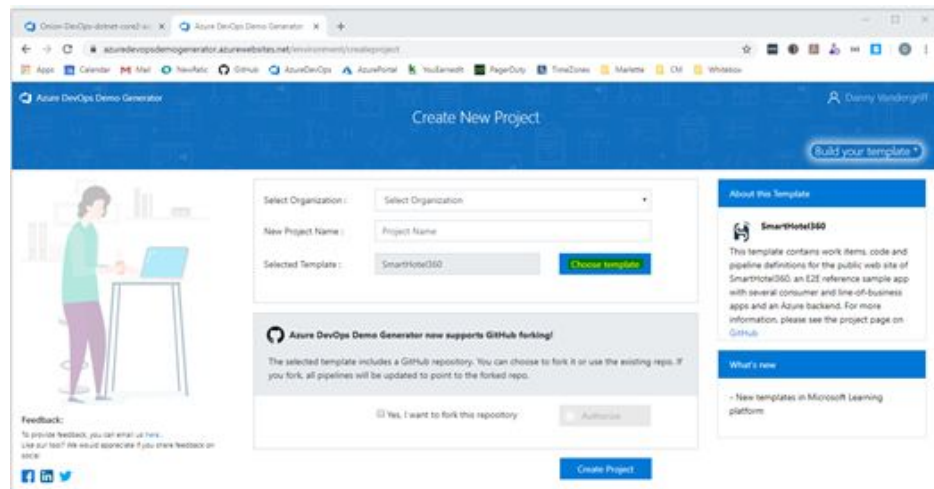
3 - Use Demo Generator Tool to Create New Azure DevOps Project from the Template

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

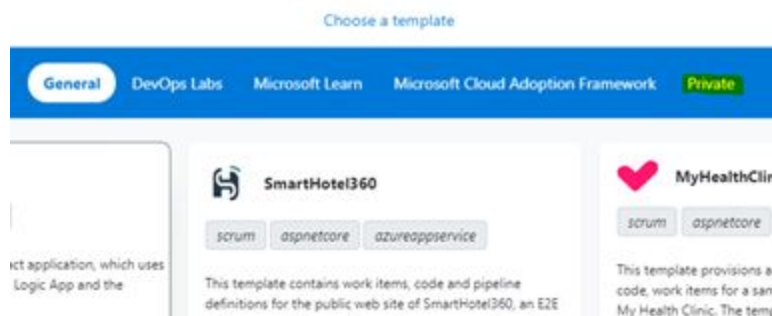
<https://azuredevopsdemogenerator.azurewebsites.net/>



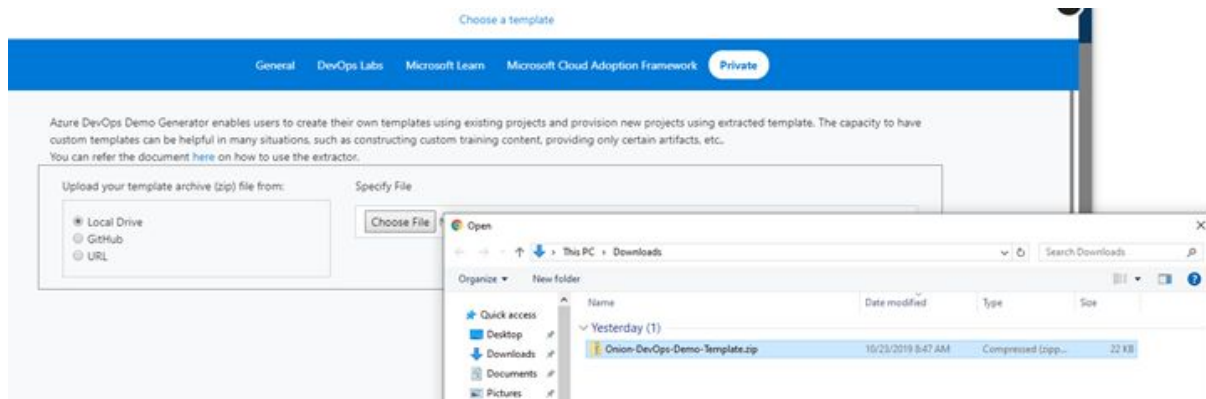
We have found this site to still be somewhat buggy, so it is important to follow the next steps in sequence. First, click the Choose Template button.



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](#)

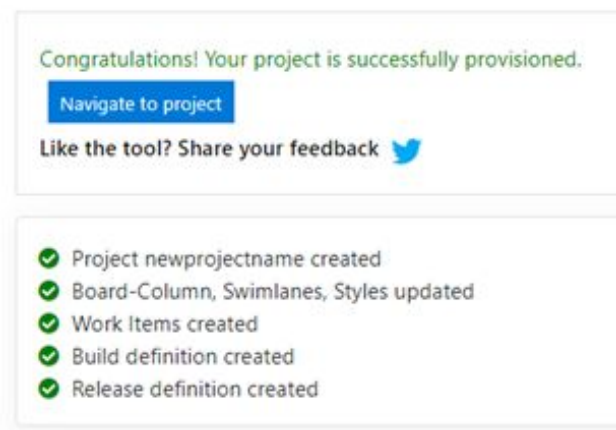
☒ The extension(s) are offered to you for your use by a third party, not Microsoft. The extension(s) is licensed separately according to its corresponding License Terms. By continuing and installing those extensions, you also agree to those License Terms.

Create Project

A status indicator will appear as the template is imported.

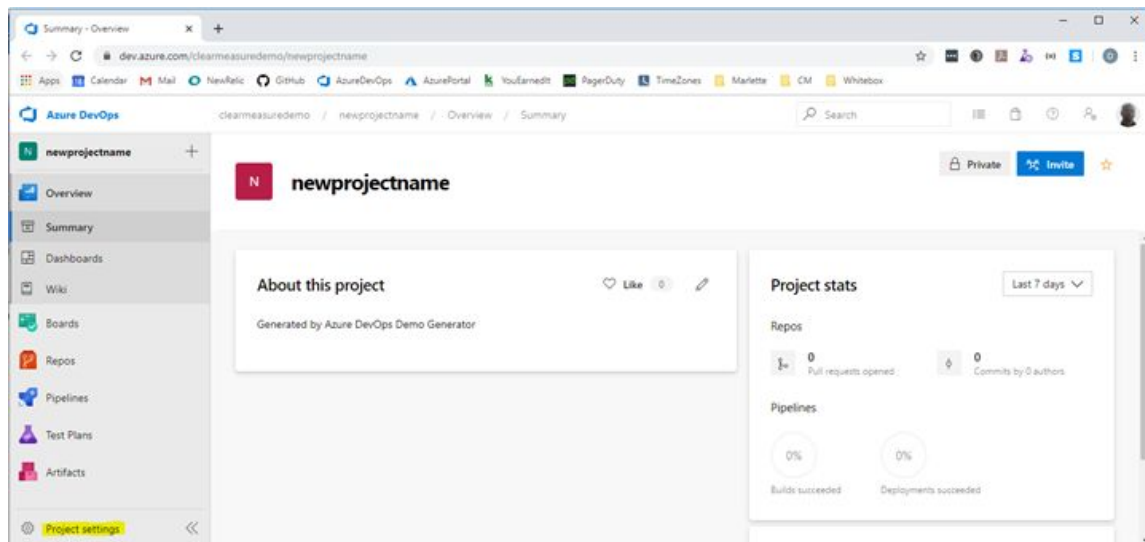


When it is complete, you will get a link to Navigate to Project. Click it and we will begin configuring your new Project.

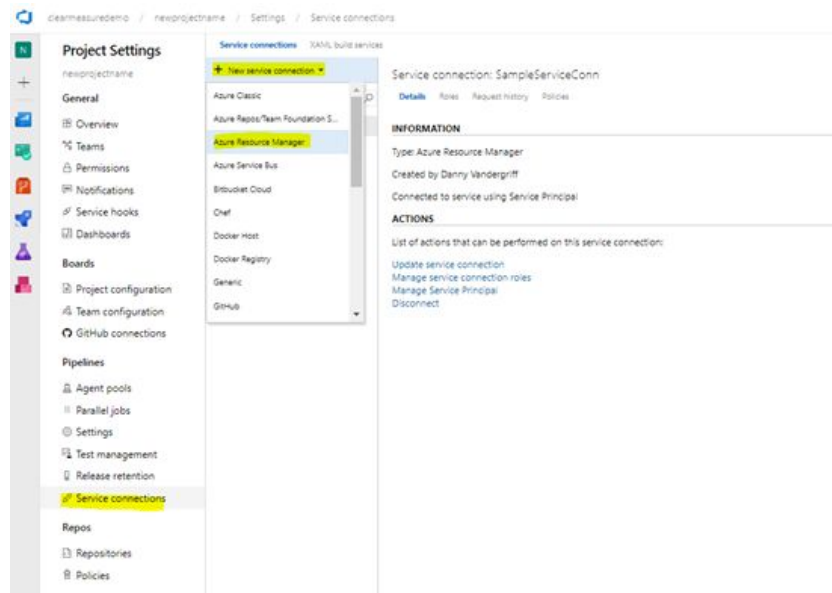


4 - Configure Project Level Settings

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.



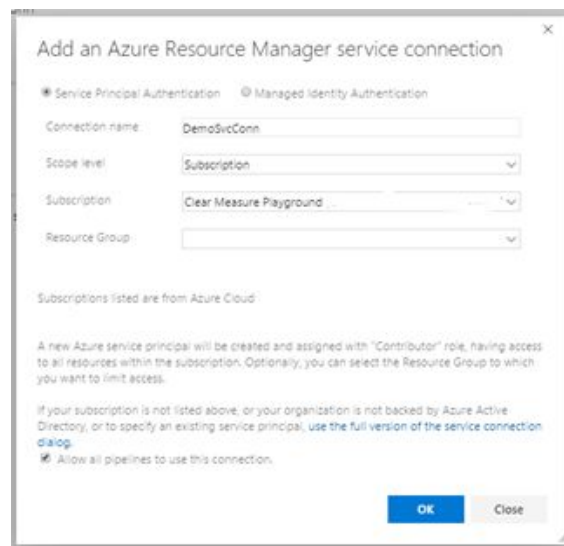
Then select Service Connections from the menu / add a New Service Connection / Azure Resource Manager. Note that there is an unconfigured SampleServiceConn, because the Demo Generator tool requires a Service Connection for the import to function. You can disregard it.



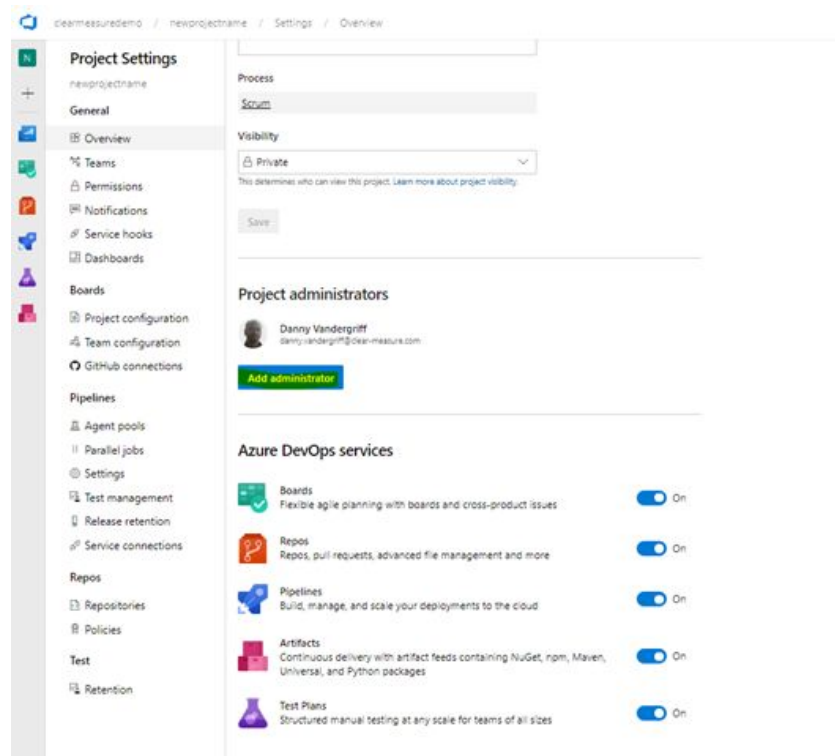
Give your new Service Connection this name: **AzureSvcConn**

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.

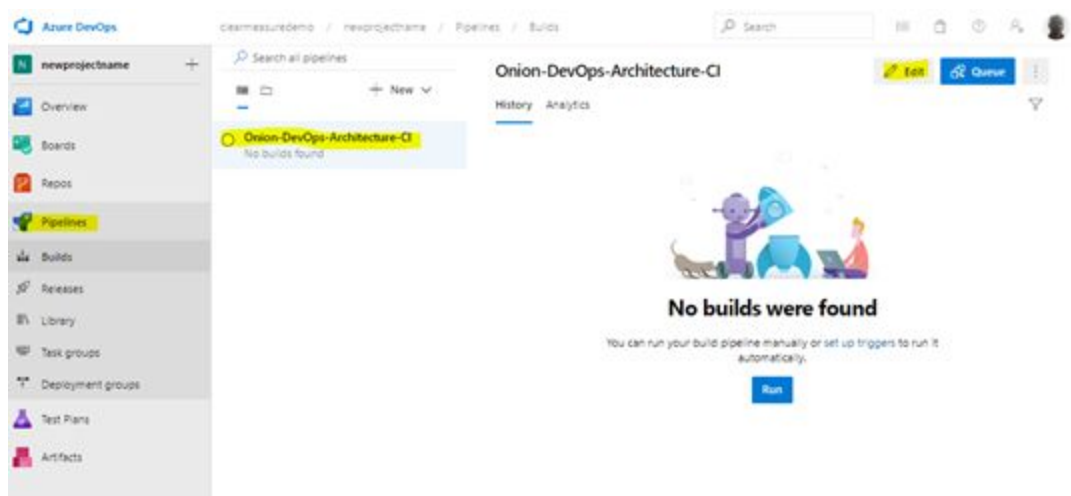


Optionally, you may add additional Project Administrators to your Organization under the Overview menu. You will need to verify that all Azure DevOps Services are set to 'On' for the demo.



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.’

The screenshot shows the Azure DevOps pipeline configuration interface. The left sidebar lists the pipeline steps, with 'Create Build Database' highlighted and marked with a red warning icon and the text 'Some settings need attention'. The main panel displays the pipeline configuration, including the name 'Onion-DevOps-Architecture-CI', the agent pool 'Hosted Windows 2019 with VS2019', and the parameters section, which indicates that the pipeline doesn't have any parameters.

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 'Create Build Database' step. The 'Azure subscription' dropdown is open, displaying a list of available Azure service connections and subscriptions. The 'DemoSvcConn' service connection is highlighted. The 'Available Azure subscriptions' list includes 'Microsoft Azure Sponsorship 1', 'DV - Visual Studio Enterprise - MPN', 'MS Azure 2019 Sponsorship', and 'Microsoft Azure Sponsorship'.

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.

This screenshot shows the configuration for the 'NuGet Push' task within an Azure DevOps pipeline. The task is part of 'Agent job 1' and is currently selected. The configuration includes:

- Task version:** 2.*
- Display name:** NuGet Push
- Command:** push
- Path to NuGet package(s) to publish:** build*. \$(Build.BuildNumber).nupkg
- Target feed location:** This organization/collection (selected)
- Target feed:** clearmeasuredemo (selected from a dropdown)
- Advanced options:** Allow duplicates to be skipped (unchecked)

Buttons at the top include 'Save & queue', 'Discard', 'Summary', 'Queue', and 'Remove'.

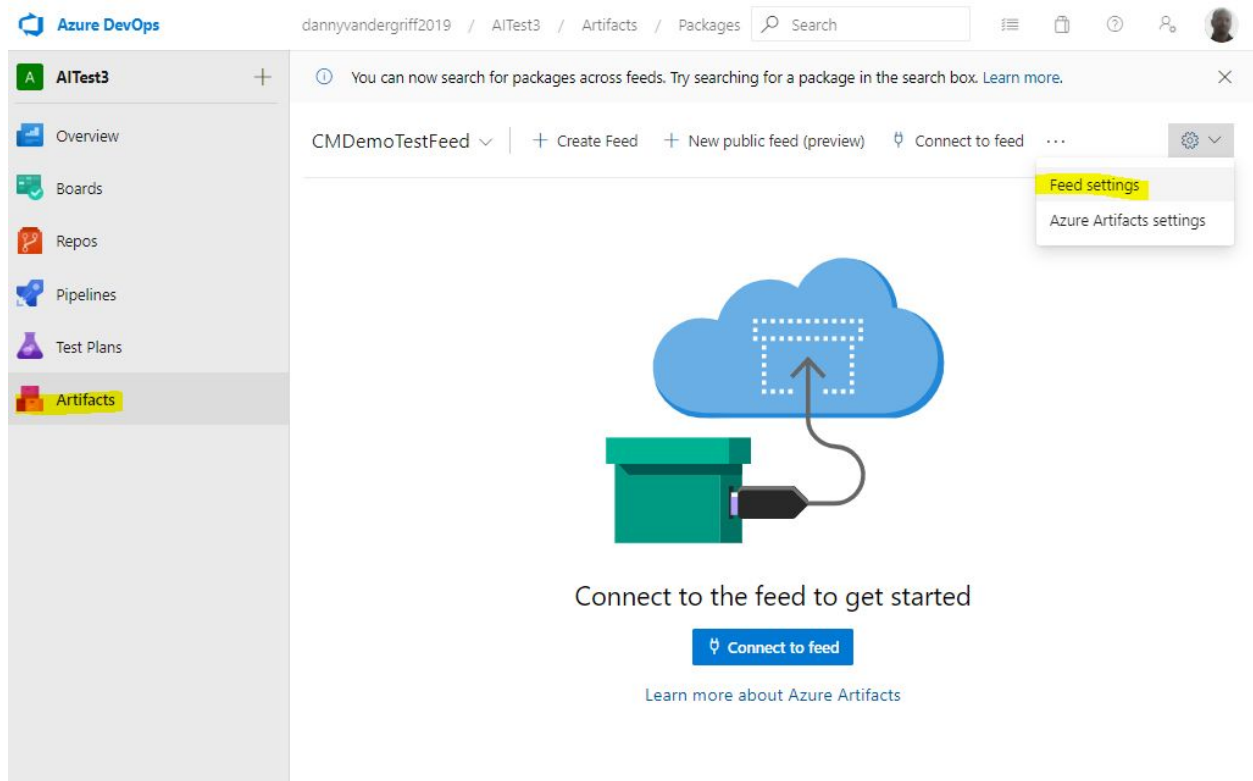
Then click the 'Save & queue' button, then 'Save and run' to initiate the first CI Build.

This screenshot shows the 'Run pipeline' dialog box, which is used to manually execute a pipeline. The dialog includes the following fields and options:

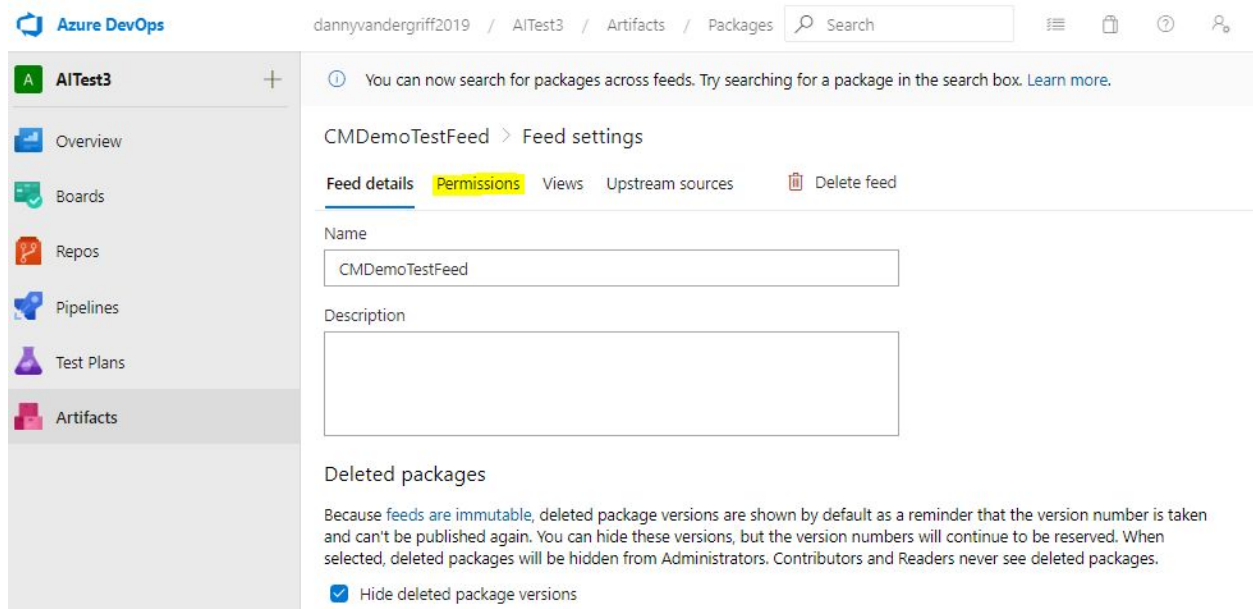
- Save comment:** A text input field for adding a comment.
- Agent pool:** Hosted Windows 2019 with VS2019 (selected)
- Branch/tag:** */ master (selected)
- Variables:** A list of variables including BuildConfiguration (Release), BuildPlatform (Any CPU), and system.debug (false).
- Demands:** A section for adding additional demands for the run.
- Buttons:** 'Cancel' and 'Save and run' (highlighted in green).

The background shows the same pipeline configuration as the previous screenshot, with the 'Save & queue' button highlighted in yellow.

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.



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 'AI/Test3' project. The left sidebar contains navigation links: Overview, Boards, Repos, Pipelines, Test Plans, and Artifacts. The main area displays the 'CMDemoTestFeed' feed settings, specifically the 'Permissions' tab. A dropdown menu is open, showing two options: 'Allow builds and releases' and 'Allow project-scoped builds', with the latter highlighted in yellow. The table below lists the permissions for various users and groups:

User/Group	Permission	Inherited
Danny Vandergriff	Allow builds and releases	
[dannyvandergriff2019]\Project Collection Administrators	Owner	✓
[AI/Test3]\Project Administrators	Owner	
Project Collection Build Service (dannyvandergriff2019)	Contributor	
AI/Test3 Build Service (dannyvandergriff2019)	Contributor	
[AI/Test3]\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.

The screenshot shows the Azure DevOps interface for the 'newprojectname' pipeline. The left sidebar contains navigation links: Overview, Boards, Repos, Pipelines, Builds, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area displays the 'Builds' tab for the 'newprojectname' pipeline. The build status is 'Succeeded'. The build log shows the following steps:

- Initialize job - succeeded (7s)
- Checkout - succeeded (8s)
- Use .NET Core sdk 3.0.100-preview3-010431 - succeeded (15s)
- Create Build Database (2m 18s)

The 'Create Build Database' step is expanded, showing the following details:

- Task: Azure resource group deployment
- Description: Deploy an Azure Resource Manager (ARM) template to a resource group and manage virtual machines
- Version: 2.107.4
- Author: Microsoft Corporation
- Help: <https://docs.microsoft.com/azure/devops/pipelines/tasks/deploy/azure-resource-group-deployment>

The log also shows the following messages:

- Checking if the following resource group exists: BuildDB--Onion-DevOps-Architecture-CI-1.0.1.0.
- Resource group exists: false.
- Creating resource group: BuildDB--Onion-DevOps-Architecture-CI-1.0.1.0
- Resource group created successfully.
- Creating deployment parameters.
- The detected encoding for file 'D:\a\1\src\BuildDatabase\database.json' is 'utf-8'
- Starting Deployment.
- Deployment name is DatabaseARM-20191024-055005-33dc

The build log also shows the following steps:

- Capture Created Database Variables - pending
- Execute Build Script - pending
- Publish Test Results \BuildTest* - pending
- NuGet Push - pending
- Cancel - Delete Build Database 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.

The screenshot displays the Azure DevOps web interface. On the left is a navigation sidebar with icons for Overview, Boards, Repos, Pipelines, Builds, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main content area shows a build pipeline run for a project named 'newprojectname'. The build is titled '#1.0.1.0: Updated chromedriver.exe' and is marked as 'Succeeded'. Below the title, it says 'Manually run today at 12:49 am by Danny Vandergriff @ CM/DemoRepo [r-master: 6c53618]'. There are tabs for 'Logs', 'Summary', and 'Tests', with 'Summary' being the active tab. The 'Agent job 1' section shows a list of 14 steps, all of which are 'succeeded'. The steps include: Prepare job, Initialize job, Checkout, Use .NET Core sdk, Create Build Database, Capture Created Database Variables, Execute Build Script, Publish Test Results, NuGet Push, Cleanup - Delete Build Resource Group, Post-job Checkout, Finalize Job, and Report build status. The total duration of the build is 9m 17s. The interface also includes a search bar at the top right and a user profile icon.

Step	Status	Duration
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	3m 29s
Publish Test Results /build/test1*	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

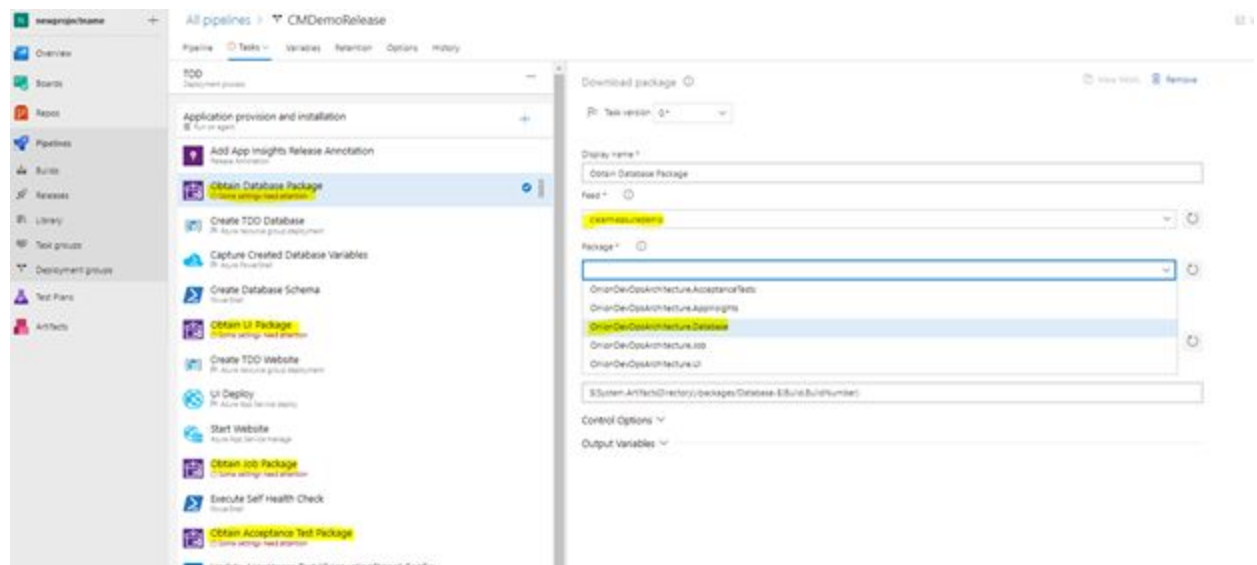
6 - Configure the CMDemoRelease Pipeline and deploy the Onion DevOps Architecture Demo application

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



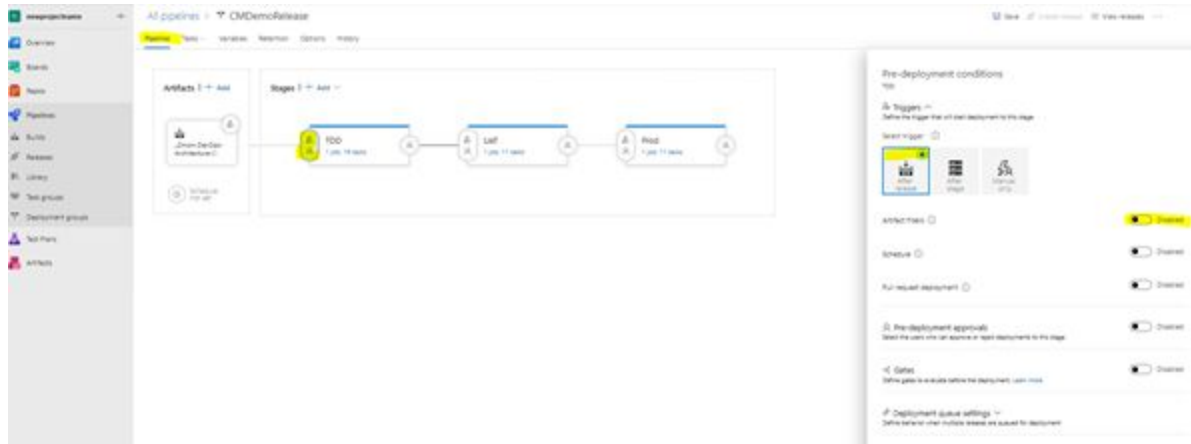
Navigate to the Tasks tab, select TDD, and note that the 'Obtain Package' steps all require attention. Each one of them will need to be updated with a Feed name and the Package name.

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.

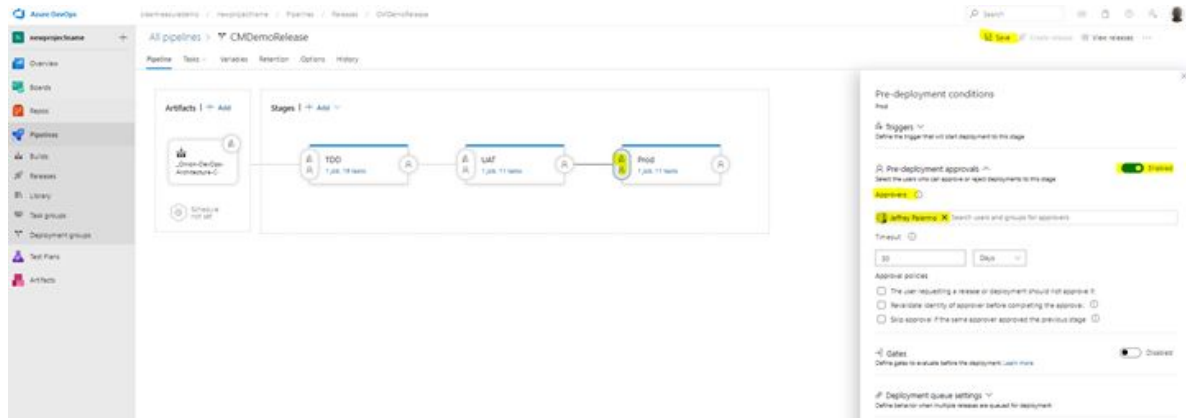
We will make the TDD Stage deploy automatically any time a CI Build creates a successful Release. Navigate to the Pipeline tab, and click the icon on the left side of the TDD Stage. Add a new Trigger for 'After Release'. We will not put any Artifact Filters on this trigger, so this configuration will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



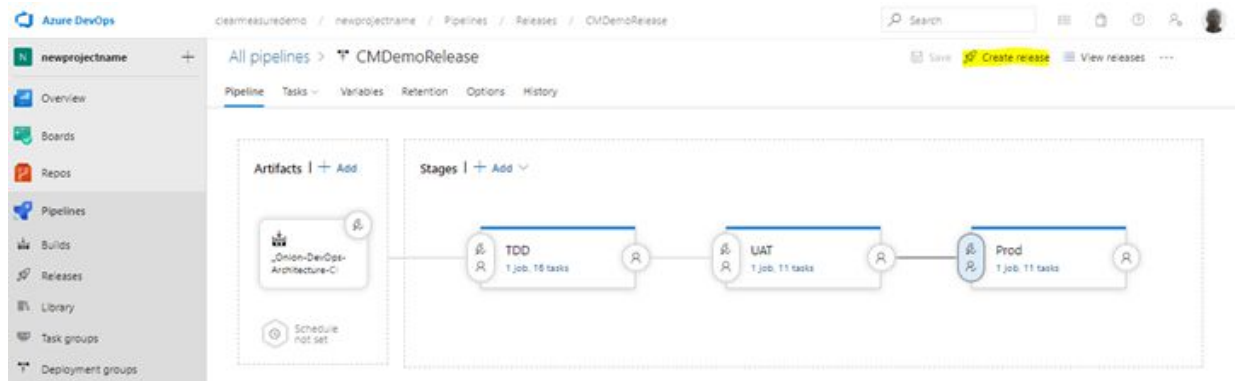
The UAT Stage has already been configured to deploy automatically after a successful TDD deployment, and it also has an Artifact Filter enabled so that only build artifacts derived from the Master branch can be released to UAT.



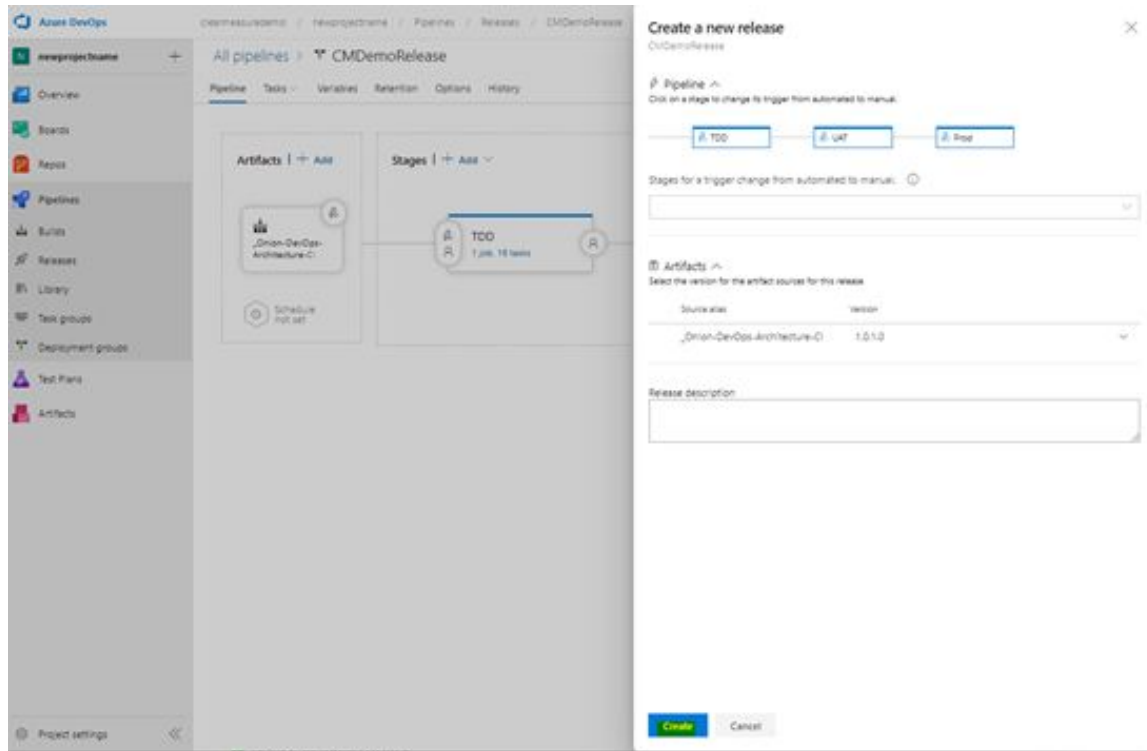
Finally, we will add a pre-deployment Approval for Production. Click the icon on the left side of the Prod Stage. Enable a pre-deployment Approval, and add one or more Approvers from your AzDO Organization. Then click Save.



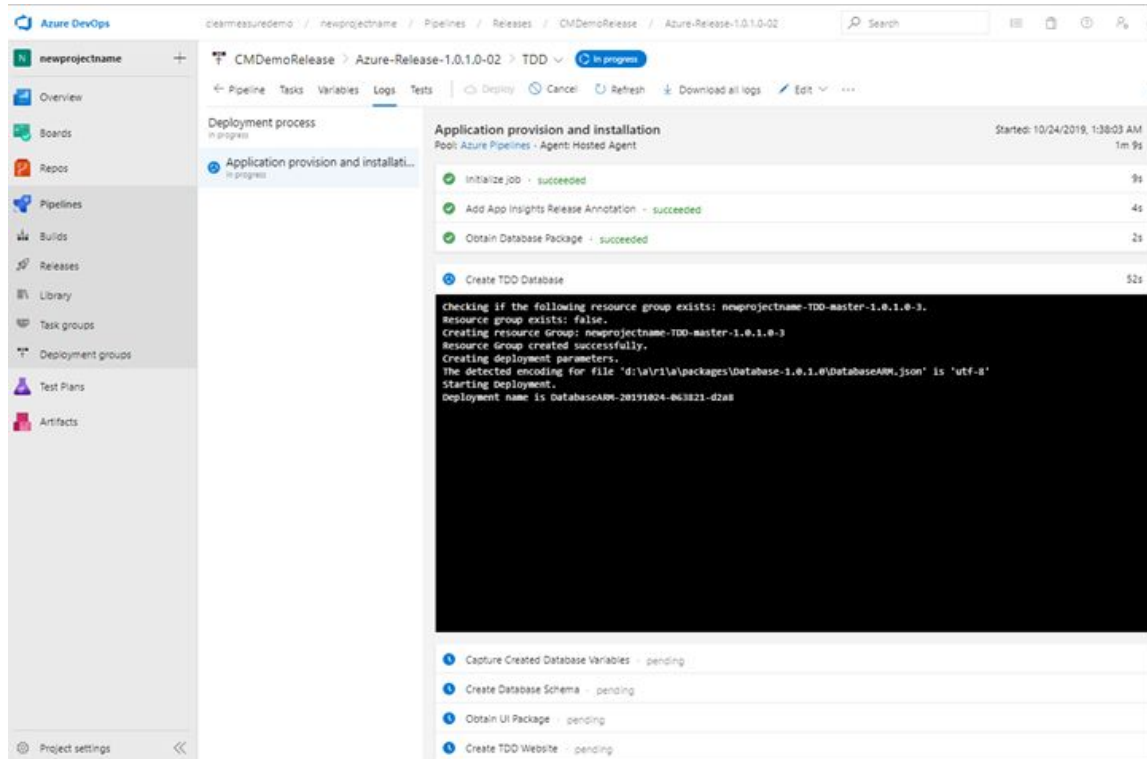
Manually create a Release to trigger a deployment to TDD.



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).

