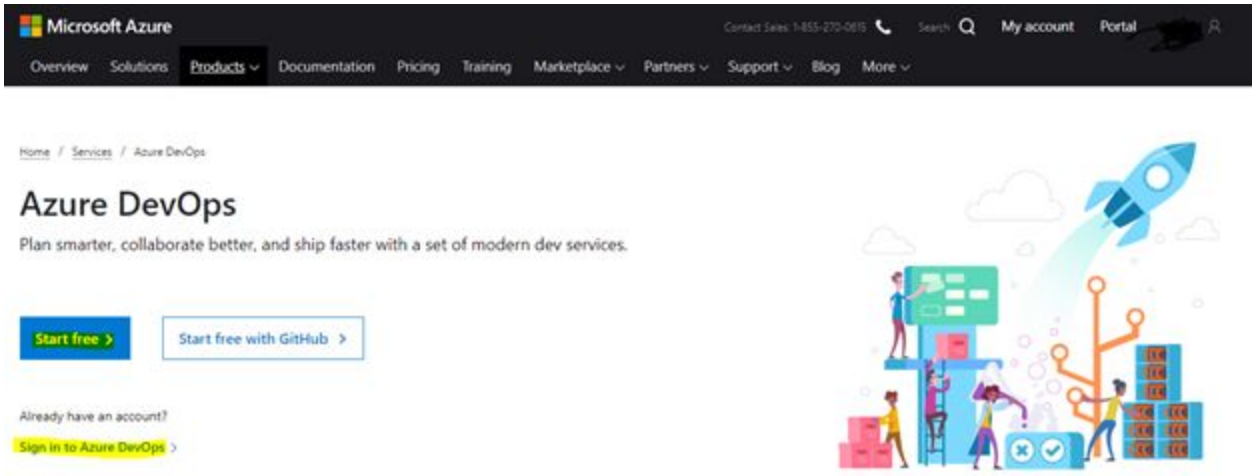
Clear Measure Onion DevOps Architecture
Azure DevOps Demo Generator Template Implementation

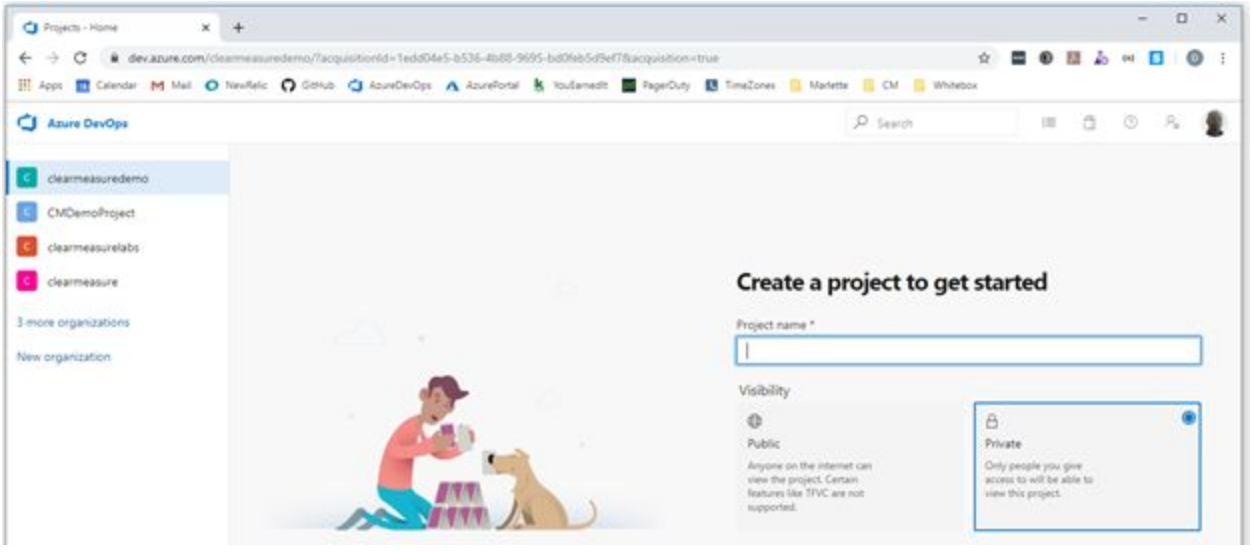
1 - Create new Azure DevOps Organization

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.

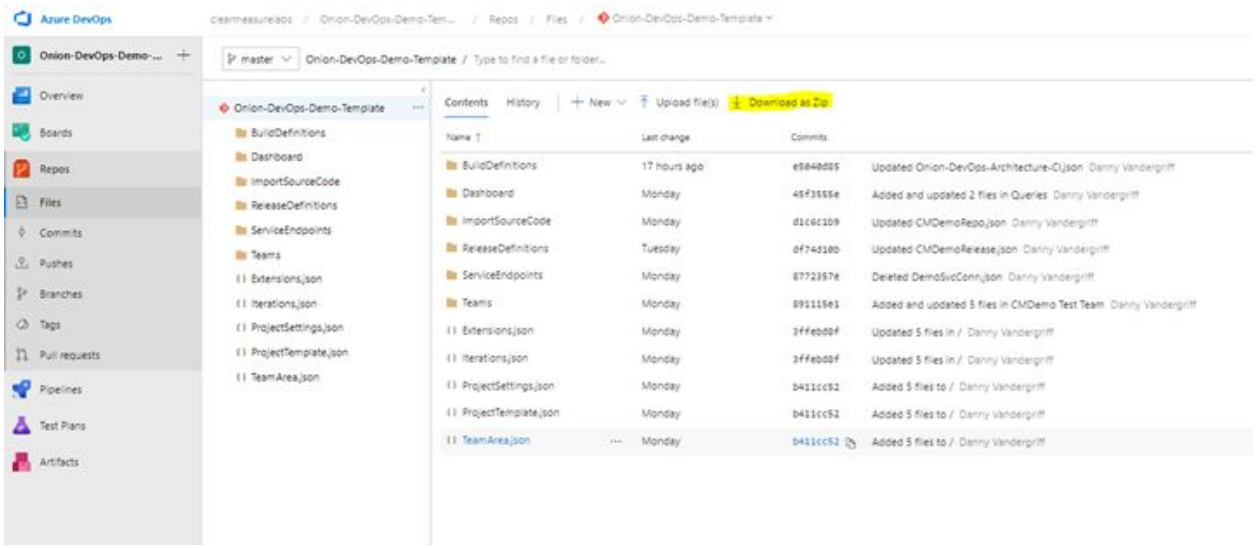


2 - Download the Clear Measure Demo Template package

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

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

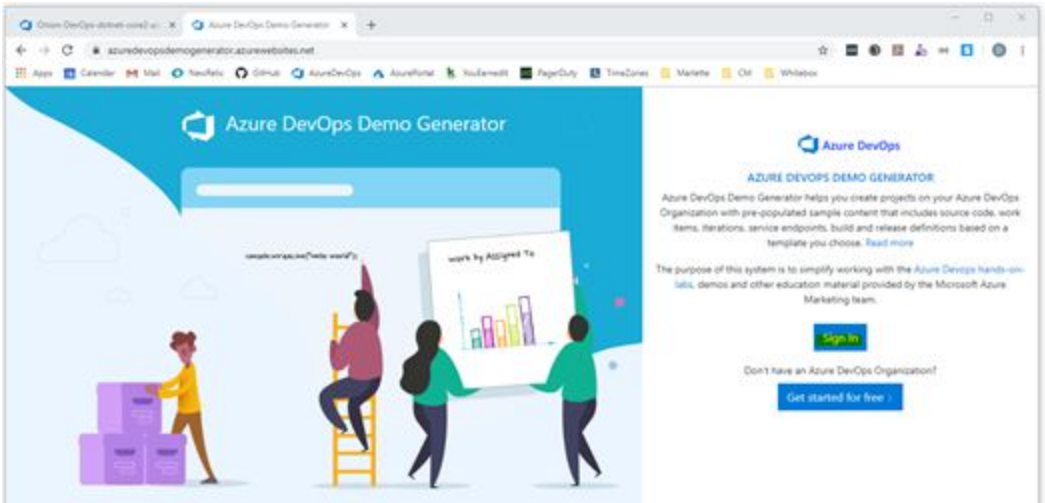
From this screen, click ‘Download as Zip’ and save the package locally.



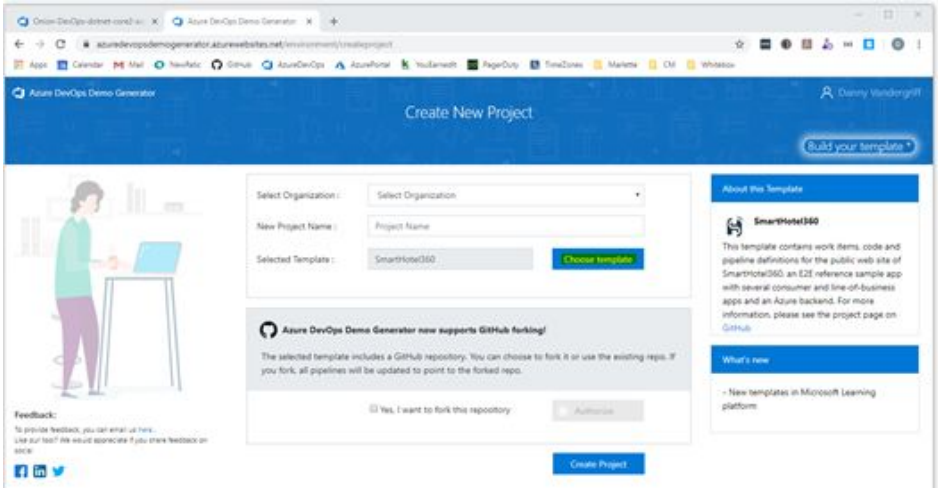
3 - Use Demo Generator Tool to Create New Azure DevOps Project from the Template

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

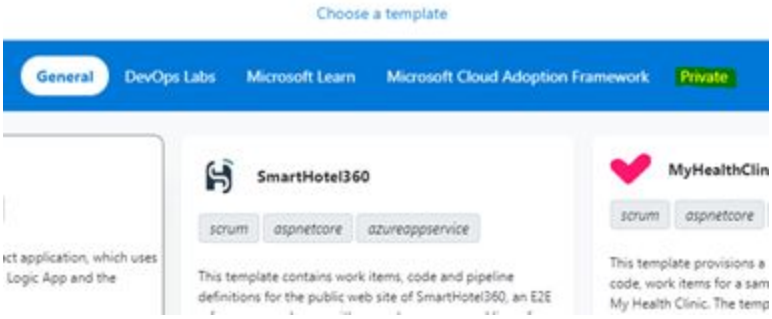
https://azuredevopsdemogenerator.azurewebsites.net/



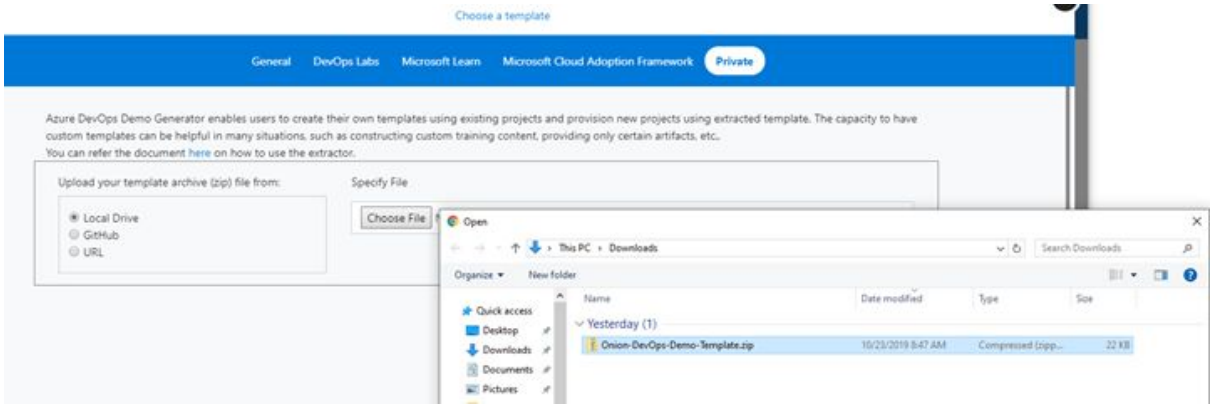
We have found this site to still be somewhat buggy, so it is important to follow the next steps in sequence. First, click the Choose Template button.



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



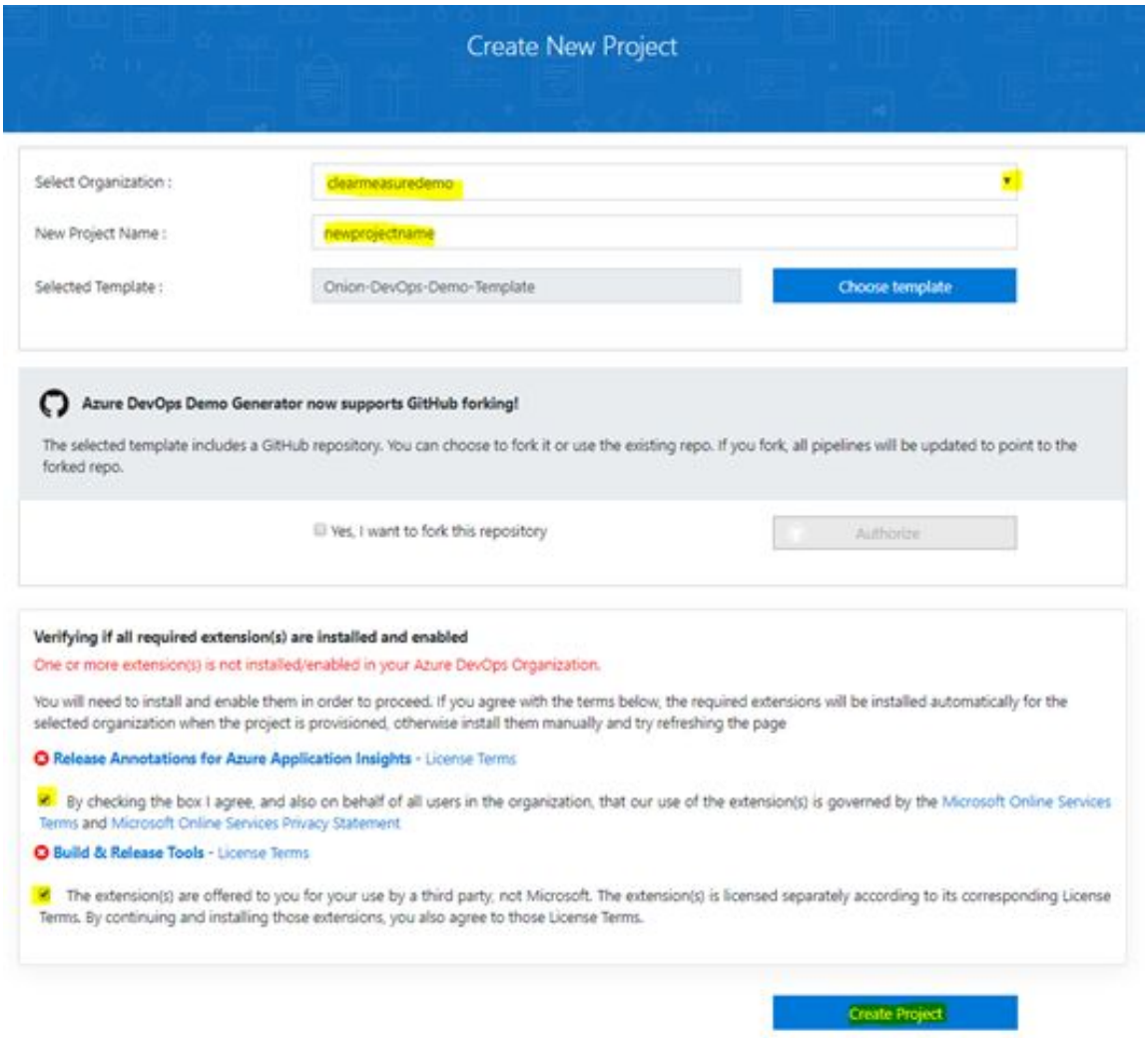
Click Choose File, then navigate to the Demo template zip file you have saved, and click Submit.



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

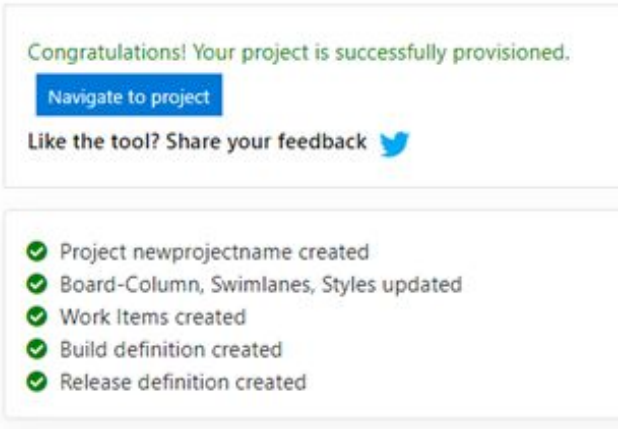
Note: Do not put underscores or any special characters in your project name, as this string will ultimately become part of an Azure app service URL.

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.



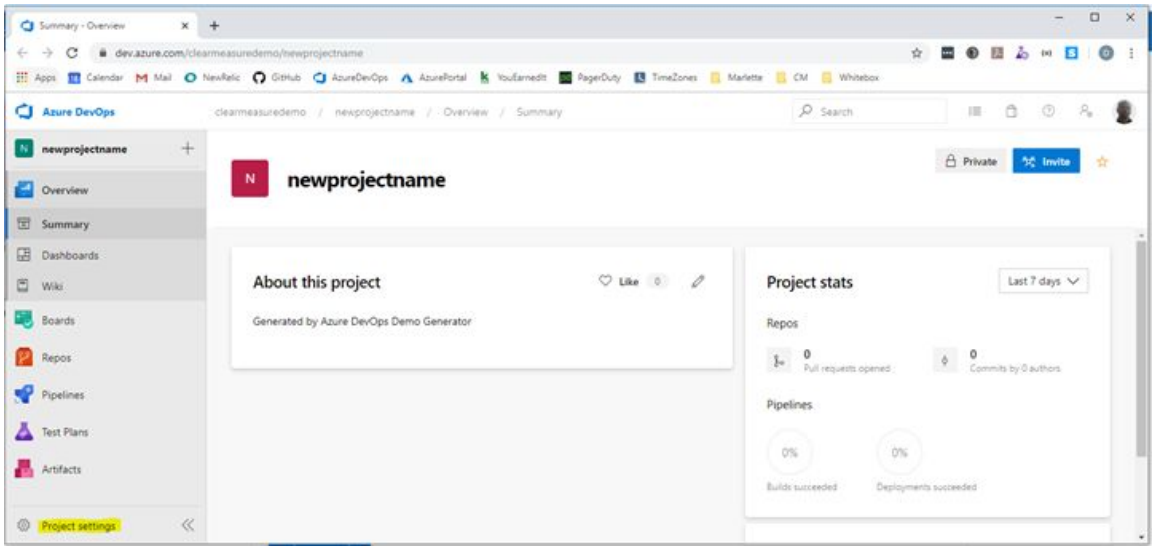
A status indicator will appear as the template is imported.

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

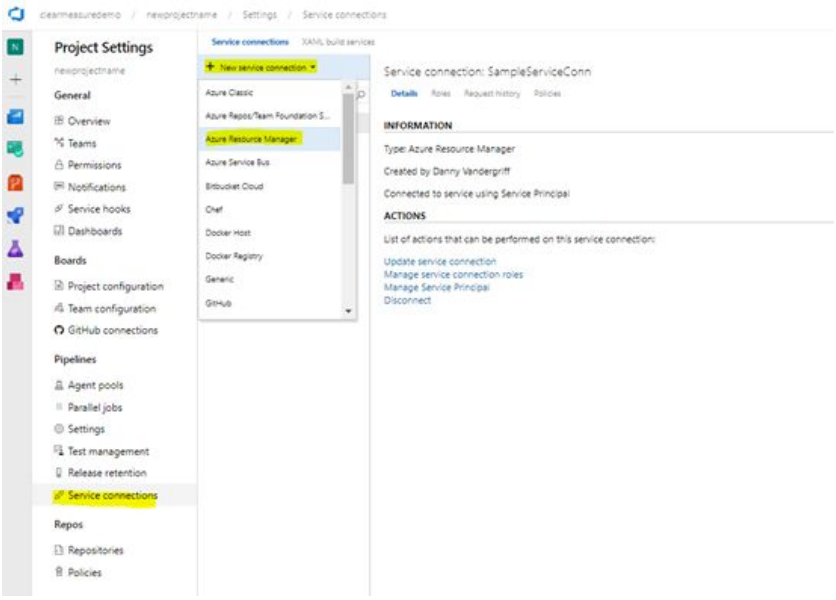


4 - Configure Project Level Settings

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.



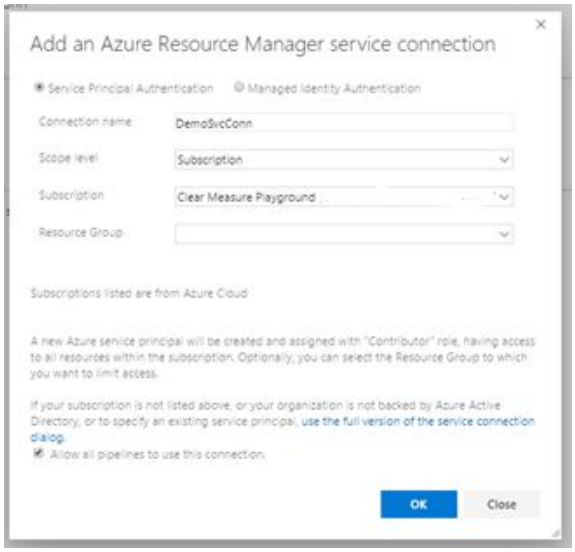
Then select Service Connections from the menu / add a New Service Connection / Azure Resource Manager. Note that there is an unconfigured SampleServiceConn, because the Demo Generator tool requires a Service Connection for the import to function. You can disregard it.



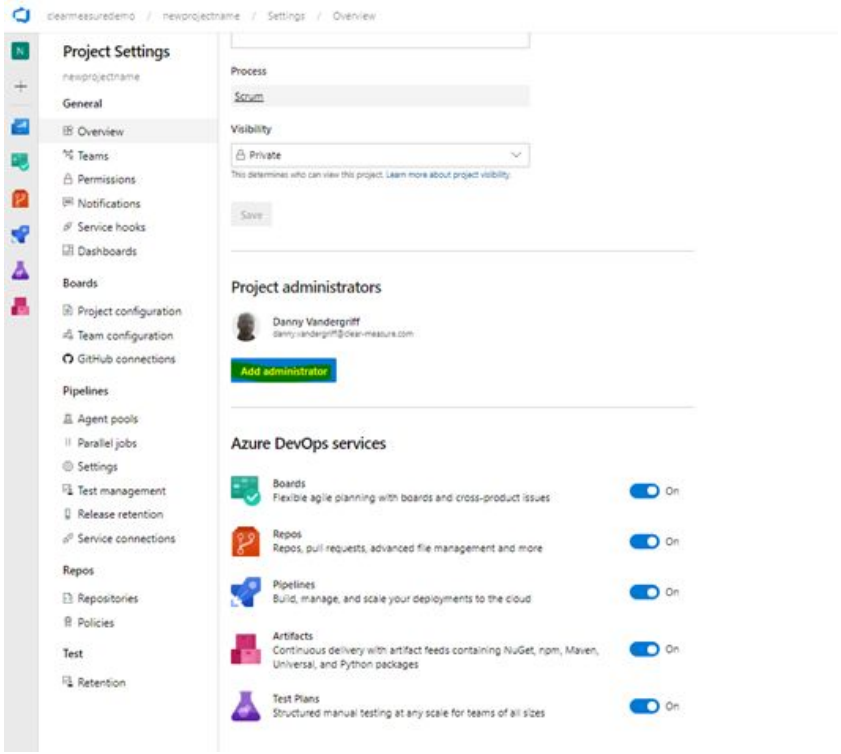
Give your new Service Connection this name: **DemoSvcConn**

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.

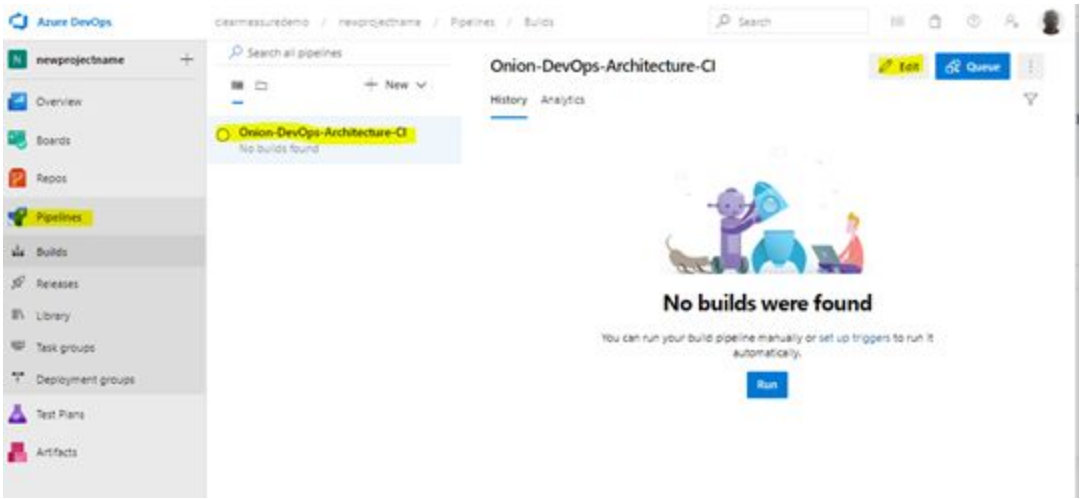


Optionally, you may add additional Project Administrators to your Organization under the Overview menu. You will need to verify that all Azure DevOps Services are set to 'On' for the demo.

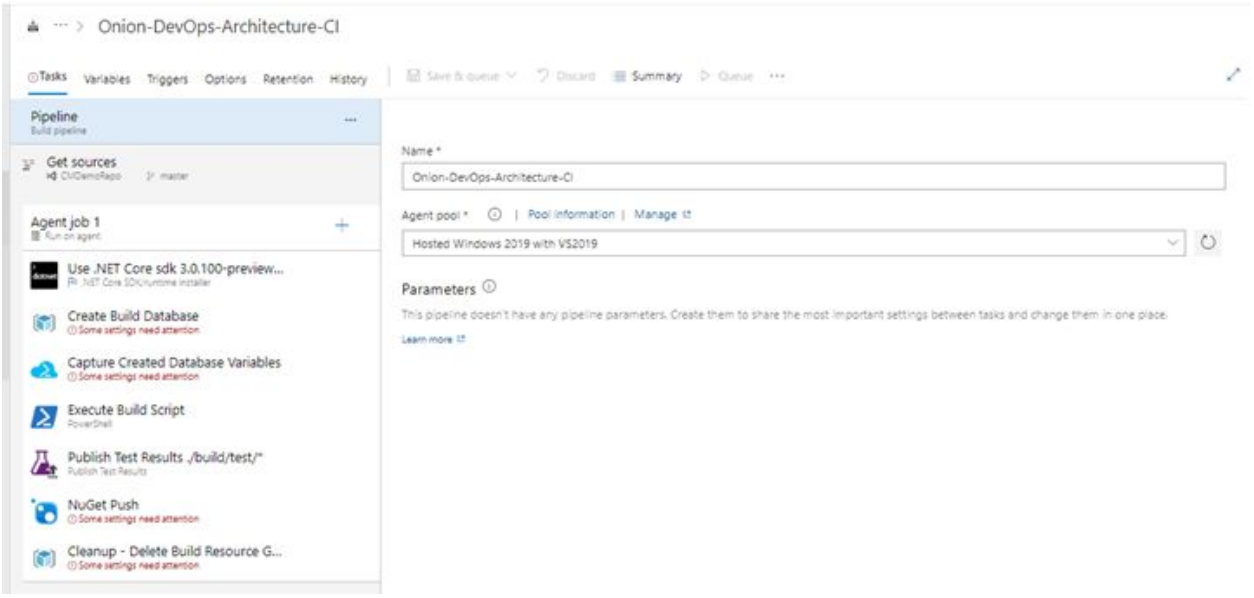


5 - Configure Onion-DevOps-Architecture CI Build Pipeline and Run Initial Continuous Integration Build

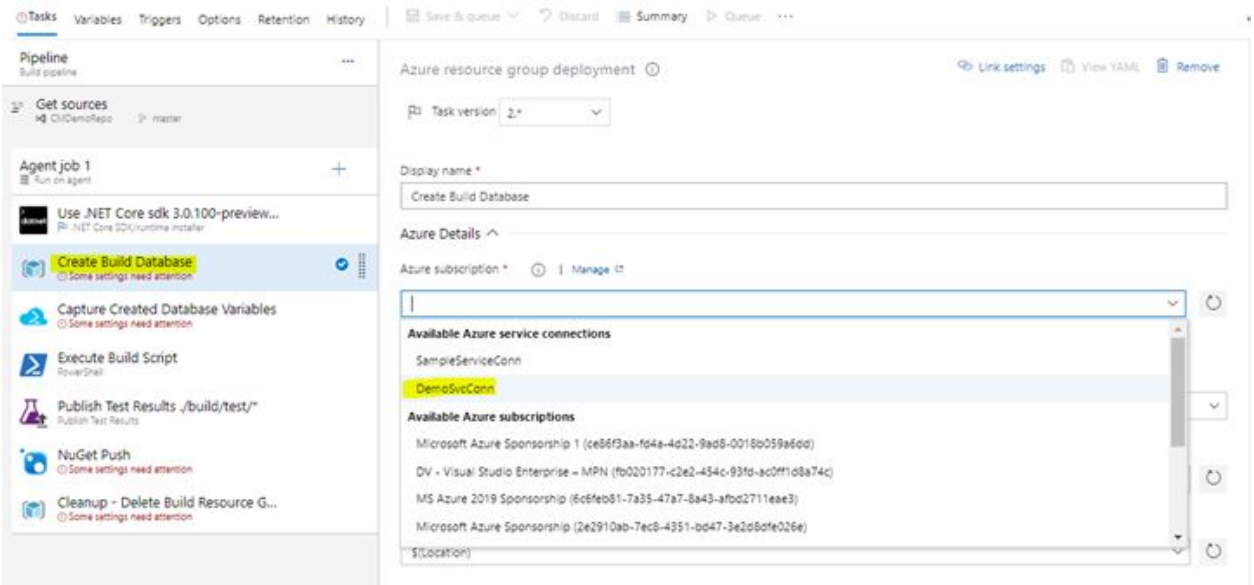
We will now begin setting up the CI Build Pipeline for the initial build, test, and package of the Onion DevOps Architecture application. Browse to Pipelines / Builds / and Edit the Onion-DevOps-Architecture-CI pipeline.



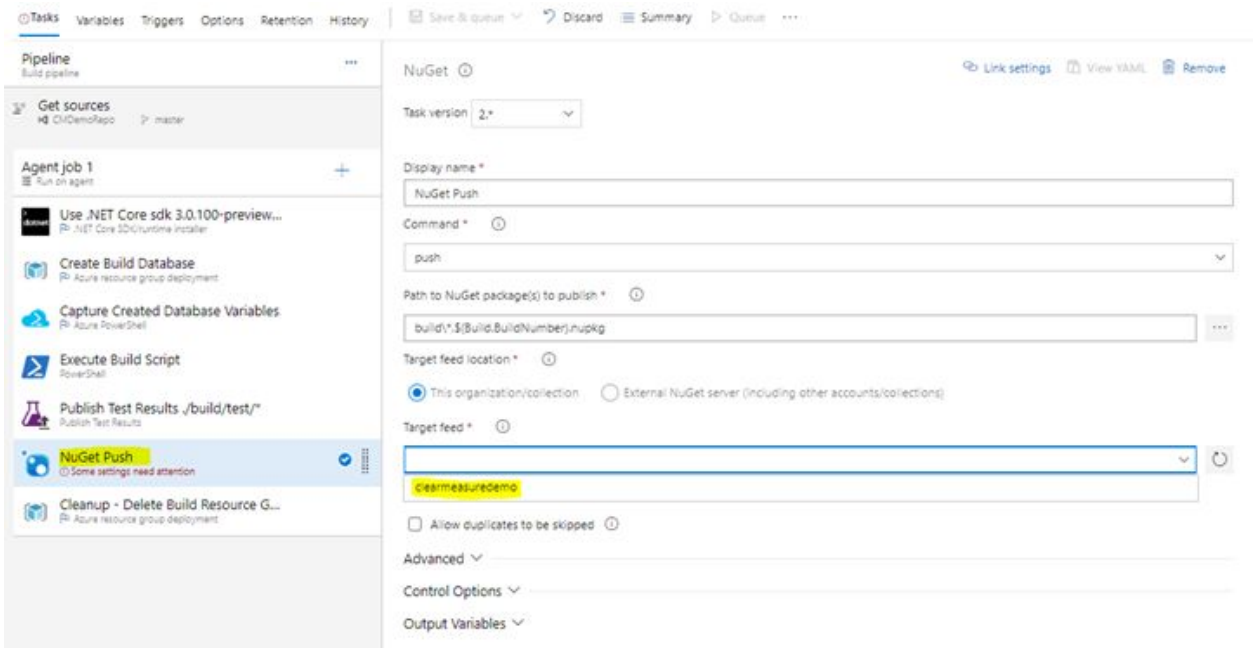
You will see the job steps listed, with four of them indicating ‘Some settings need attention.’



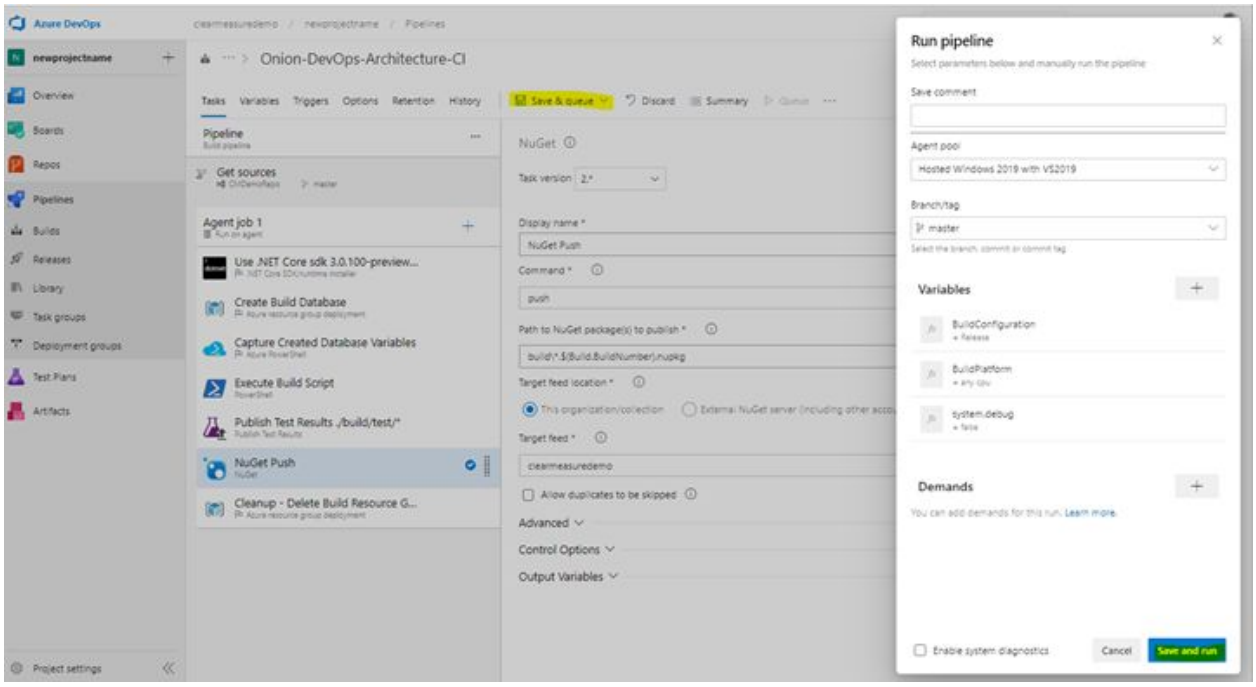
Click each step requiring attention. For the steps that require the ‘Azure Subscription’ to be populated, use the dropdown to locate the DemoSvcConn Service Connection that you created earlier.



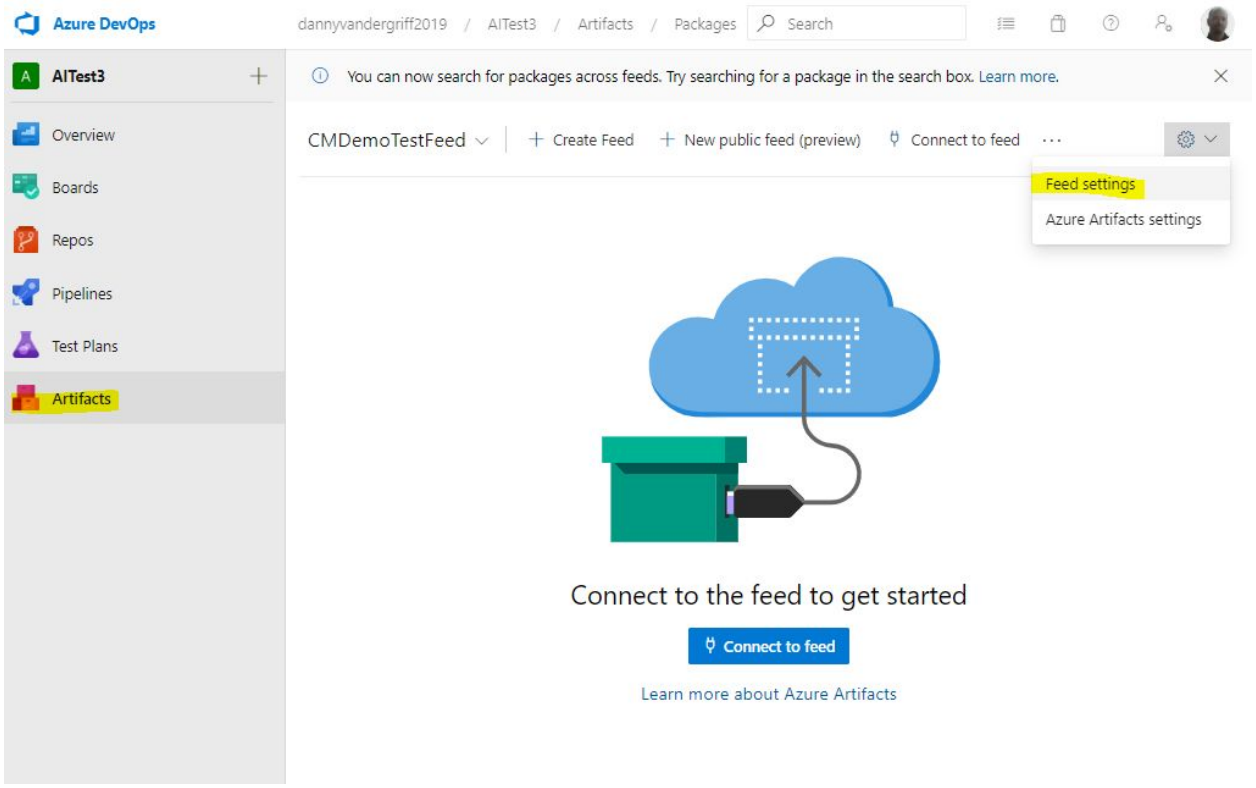
The NuGet Push step will need to have a Target Feed defined. The default Feed is created with the AzDO Organization, and shares the name. Select it for the Demo.



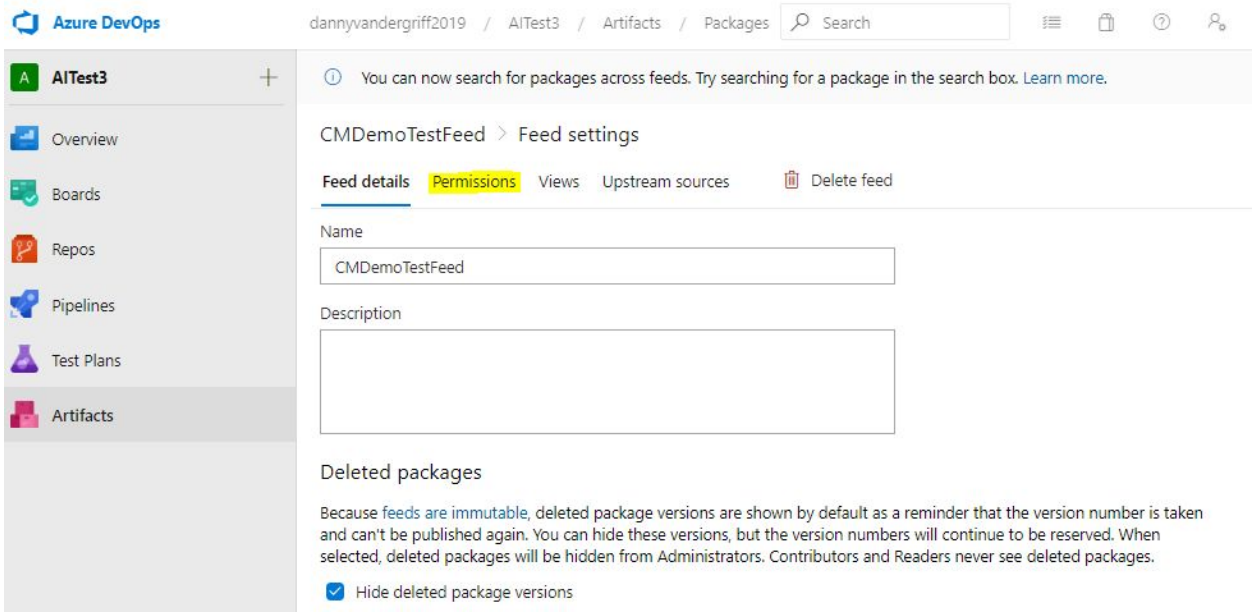
Then click the ‘Save & queue’ button, then ‘Save and run’ to initiate the first CI Build.



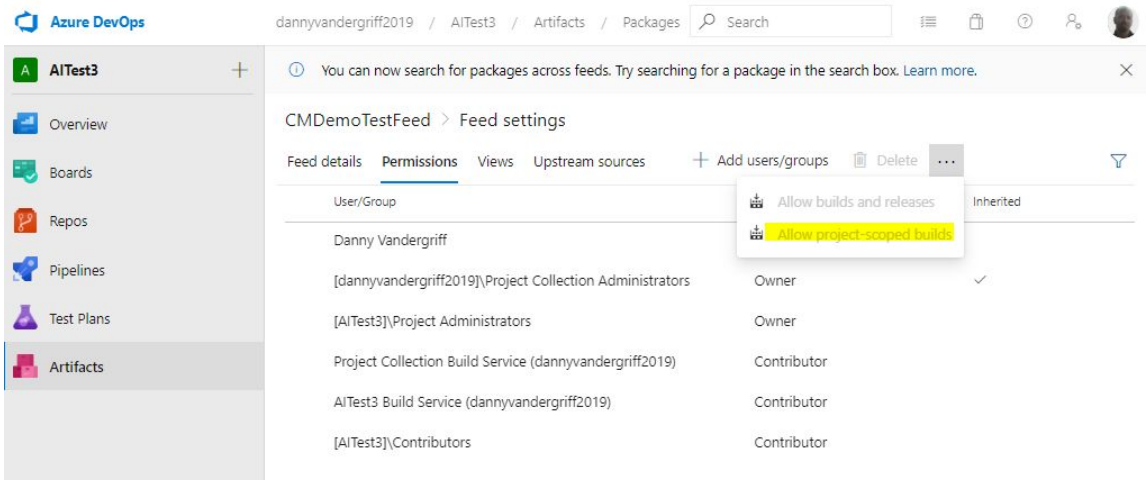
Immediately, you will need to return to Artifacts to reconfigure the Feed for your project. Azure DevOps has a reported bug where an option to allow builds to push packages into feeds is not available - until the build has been triggered! First navigate to Artifacts / Feed Settings.



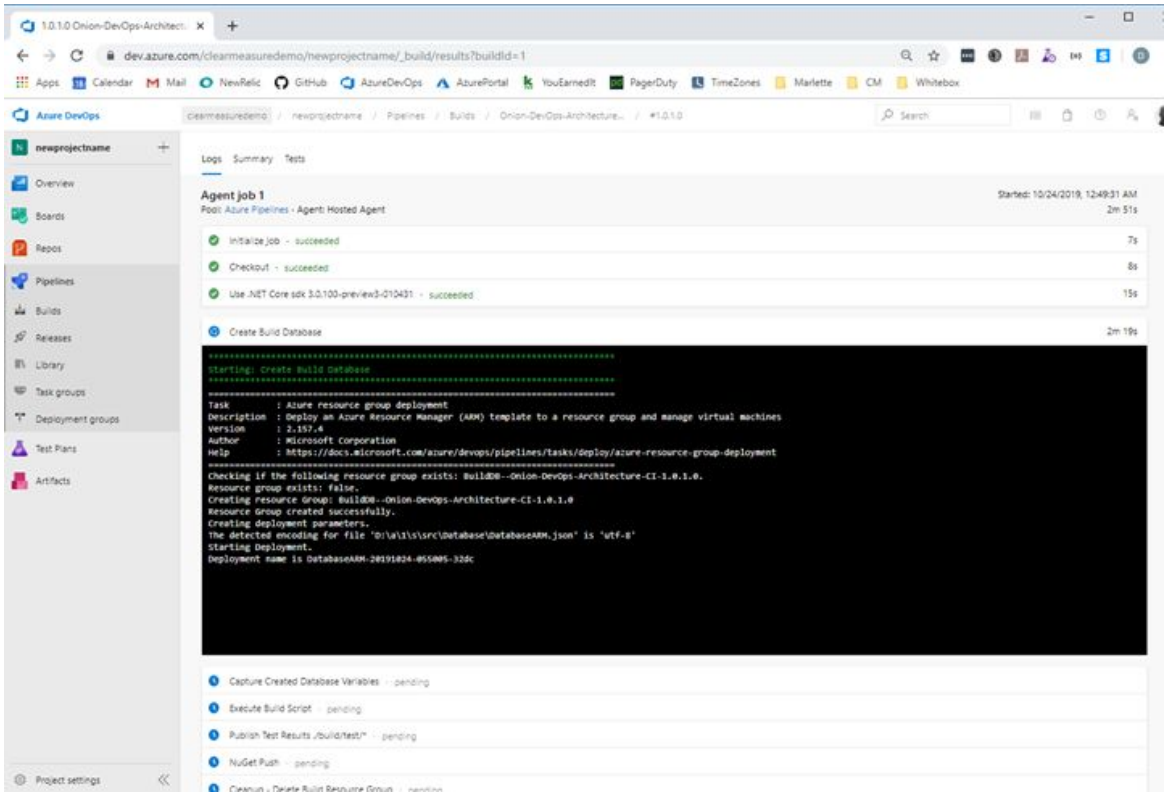
Select Permissions.



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

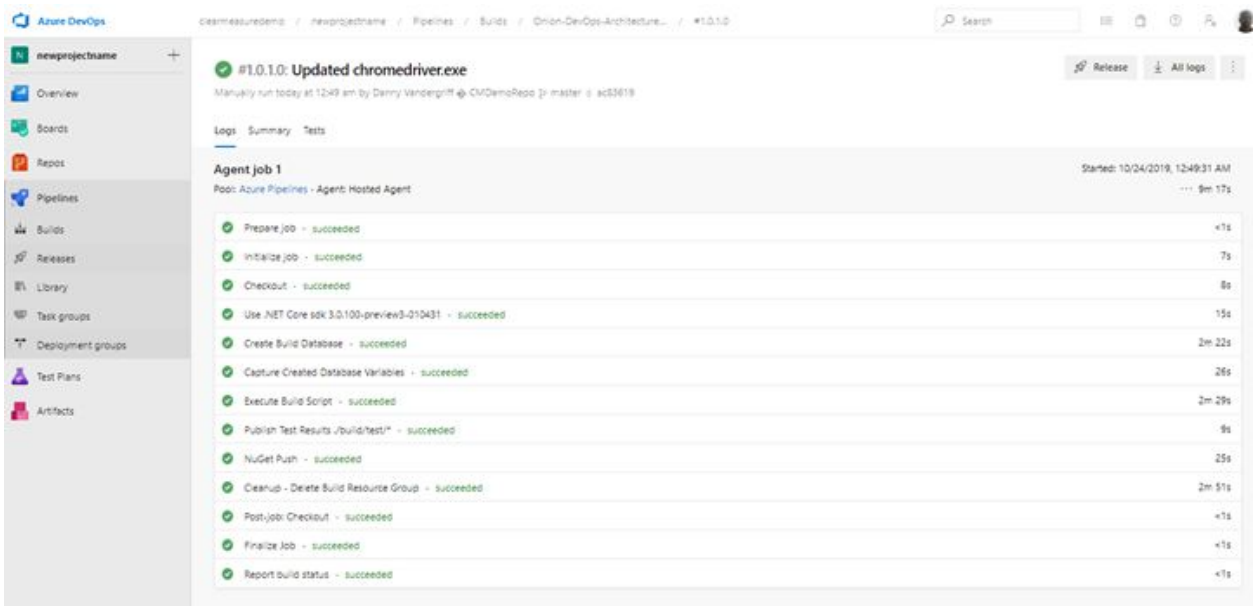


Return to the Pipeline to monitor the status of your Build. If the build has failed at the “NuGet Push” step, it is likely due to the missing permission noted above. Address the issue with the Feed, and Queue a new Build.



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.

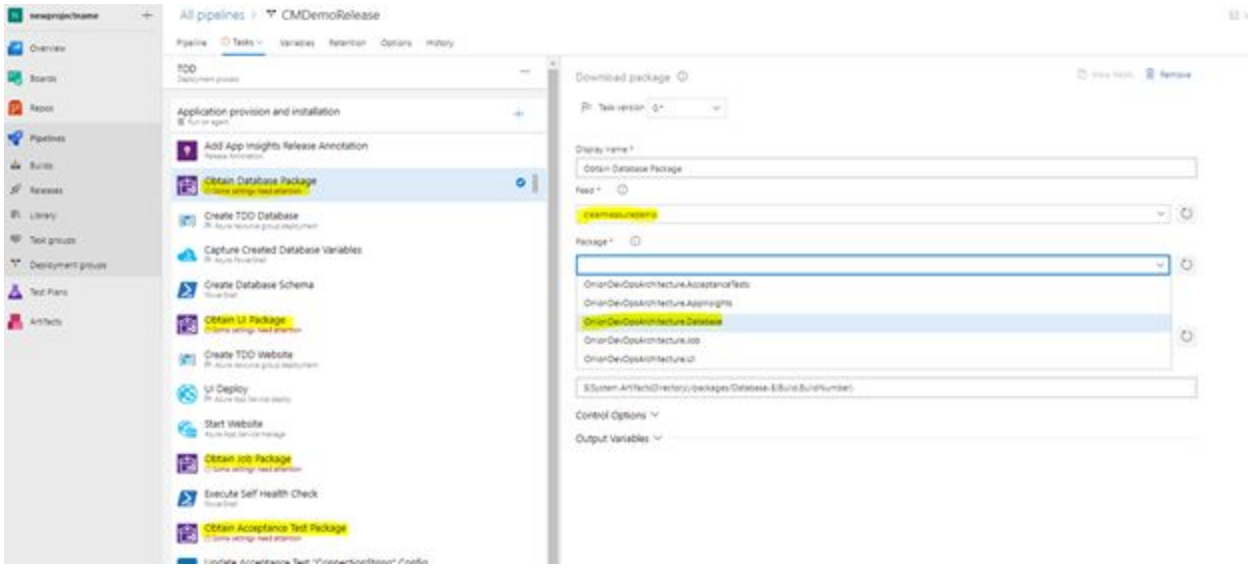


6 - Configure the CMDemoRelease Pipeline and deploy the Onion DevOps Architecture Demo application

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.



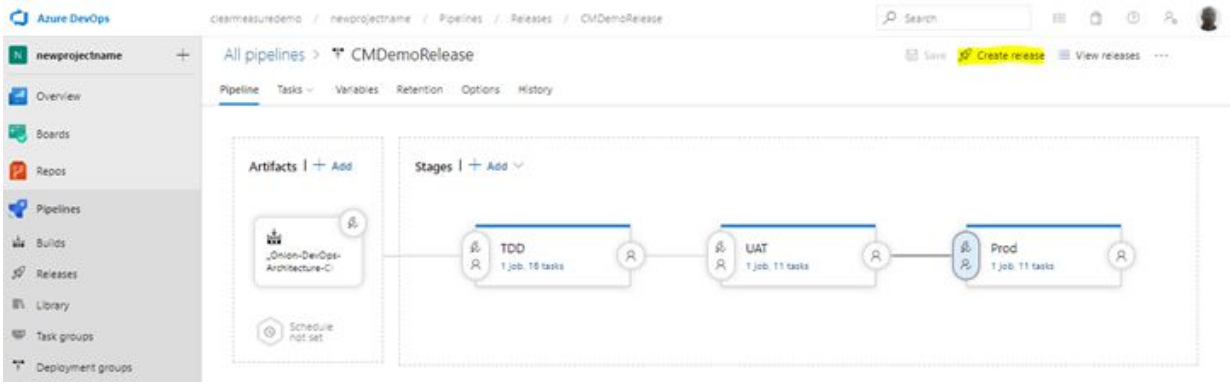
Update the Feed and Package for every Release steps that need attention in the TDD, UAT, and Prod Stages.

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.

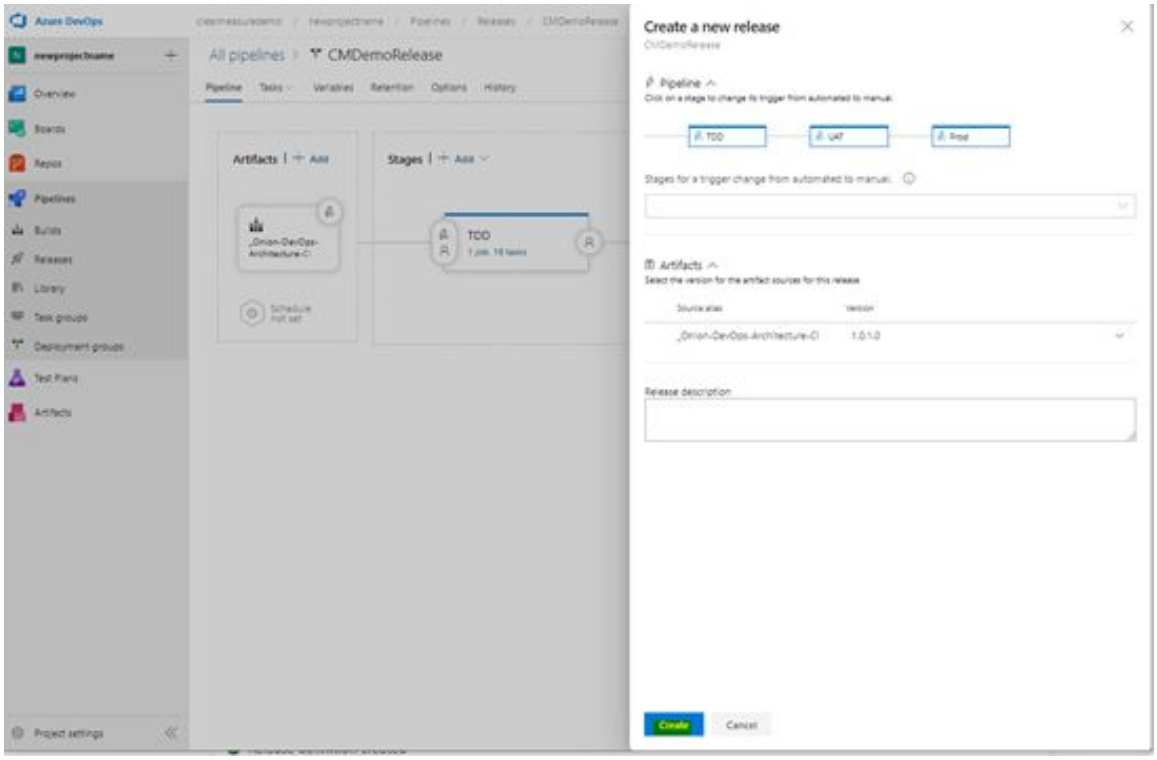
The UAT Stage has already been configured to deploy automatically after a successful TDD deployment, and it also has an Artifact Filter enabled so that only build artifacts derived from the Master branch can be released to UAT.

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.

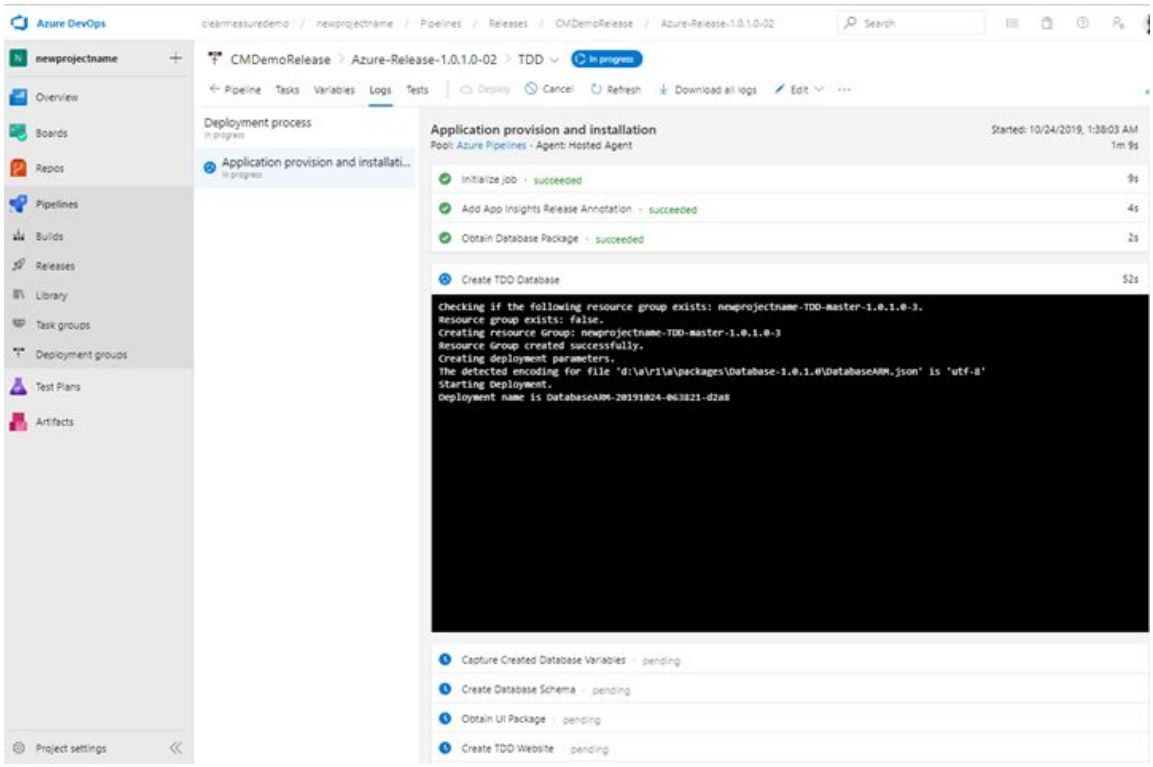
Manually create a Release to trigger a deployment to TDD.



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).