PowerApps

Power Platform Build Tools Hands-on Lab

Module 2

Automating Solution Deployment Using Azure DevOps

Last Updated: February 15, 2023

Authors: Per Mikkelsen, Shan McArthur, Evan Chaki

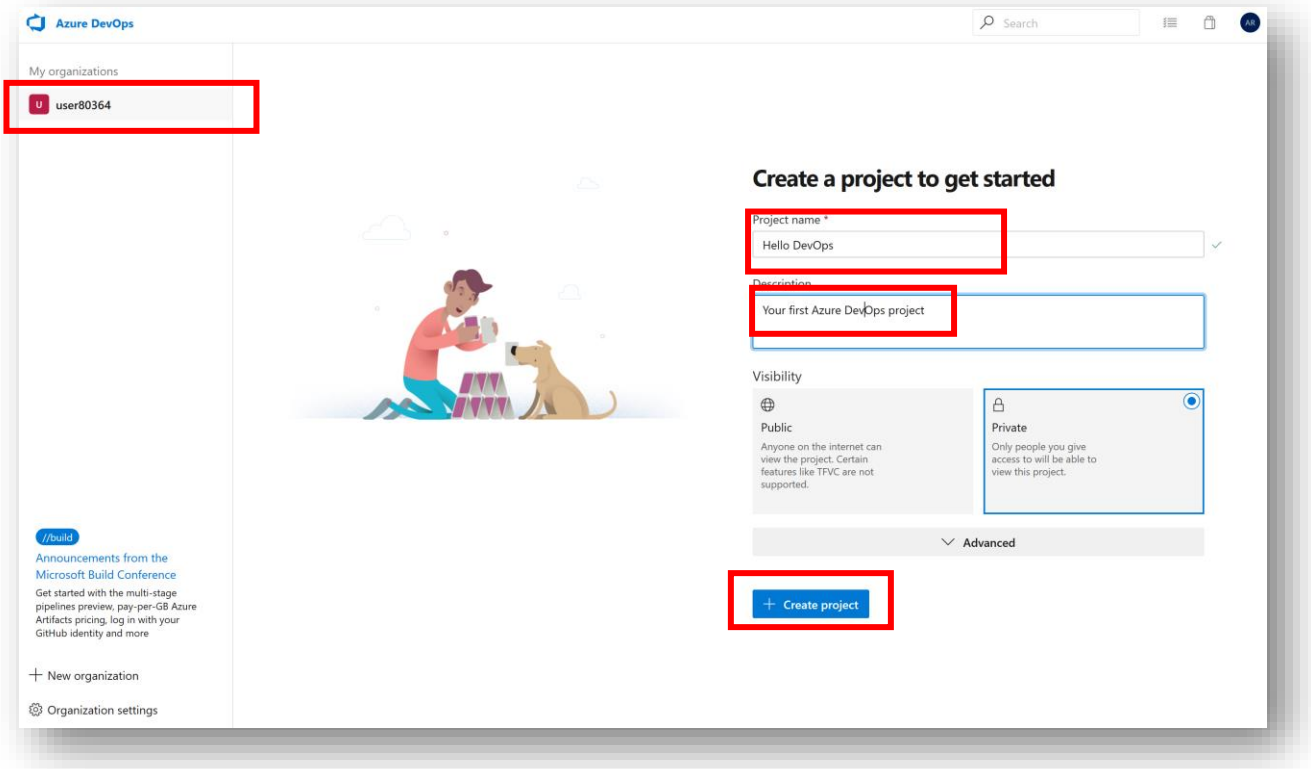
Lab Scenario

In this hands-on lab, you will first create a Microsoft Azure DevOps project, setup permissions, and create the required pipelines to automatically export your app (as an unmanaged solution) from a development environment. Next, you will generate a build artifact (managed solution). Finally, you will deploy the app into production. The lab will also introduce you to Microsoft Power Platform Build Tools.

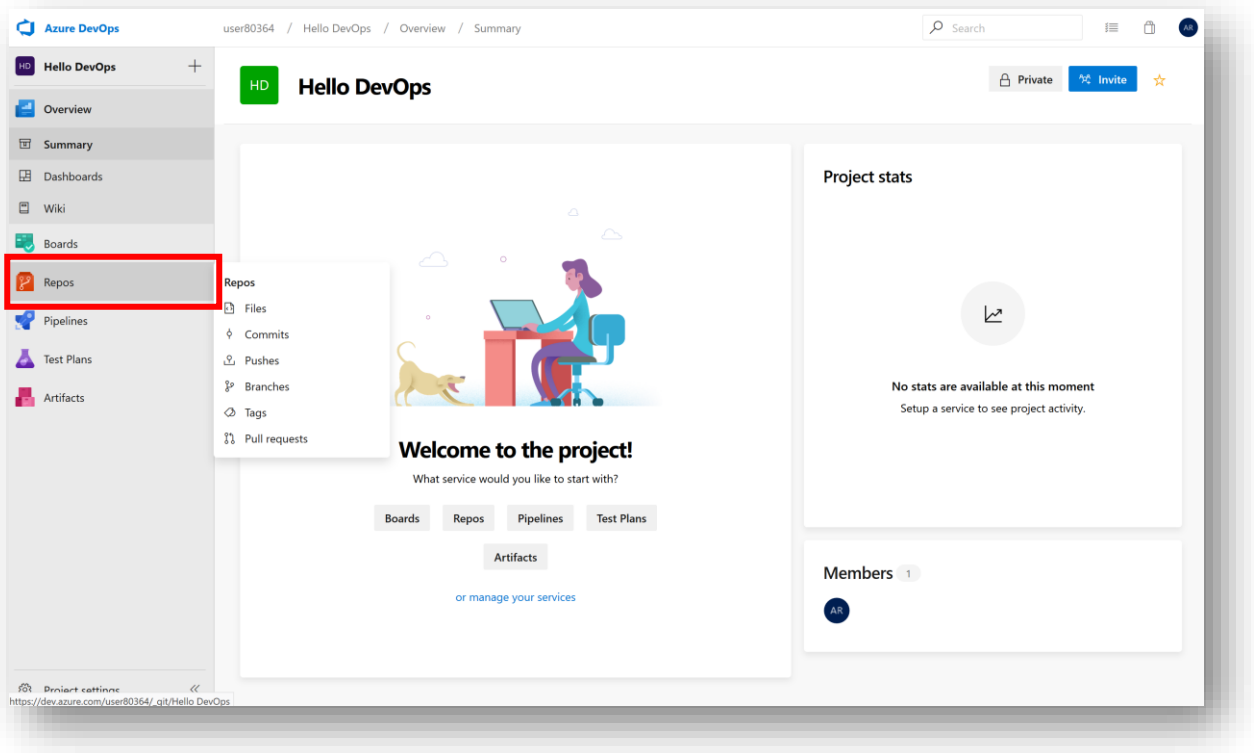
Create an Azure DevOps Project

We are going to use Azure DevOps for both source code storage and build and deployment automation. You can use any automated source control and build automation tools using the same principles. Your configuration data should be exported from a development environment, processed by the Package Deployer tool, and checked into source control.

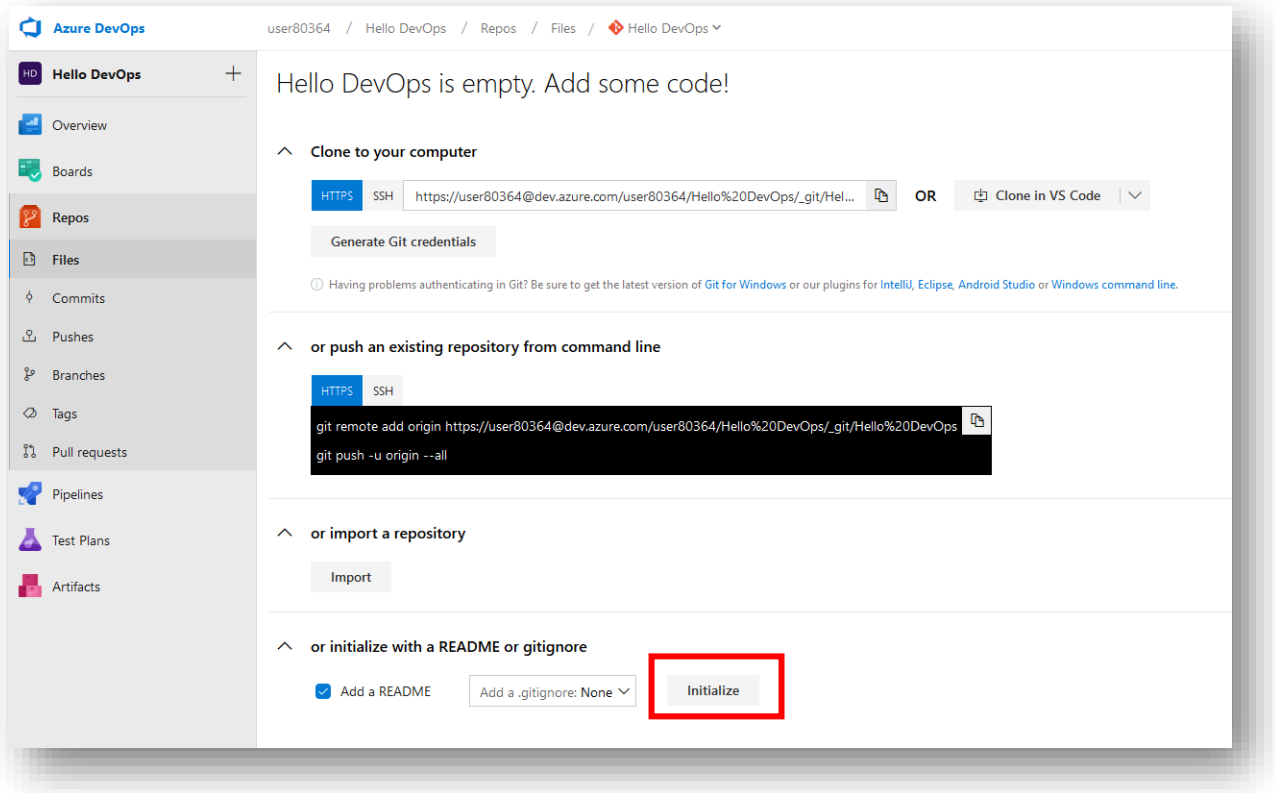
1. Log into dev.azure.com with your credentials and click on your organization in the left panel. Follow the instructions to create a project. Click **Create project**.



2. When the project is completed, you will need to create a repository to hold the source code. Click on the **Repos** link in the left navigation.

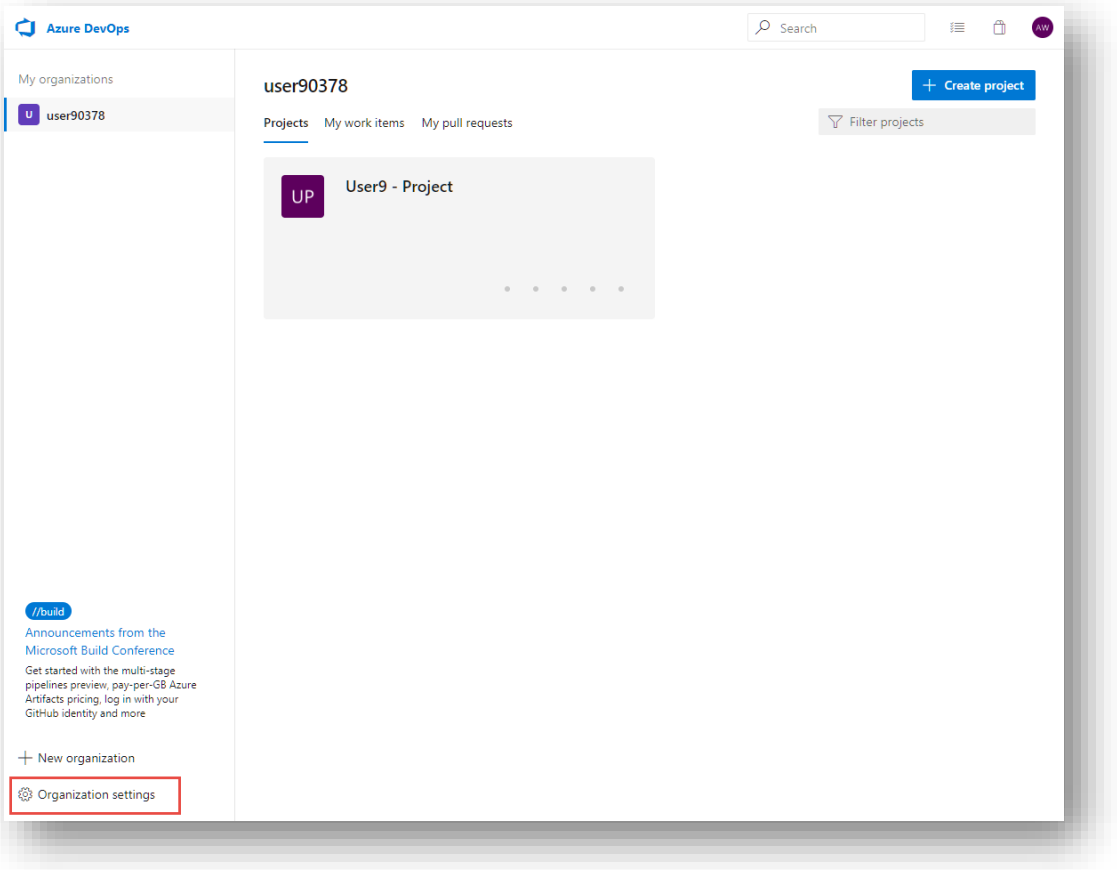


3. Initialize the repo with the default README file by clicking the **Initialize** button.

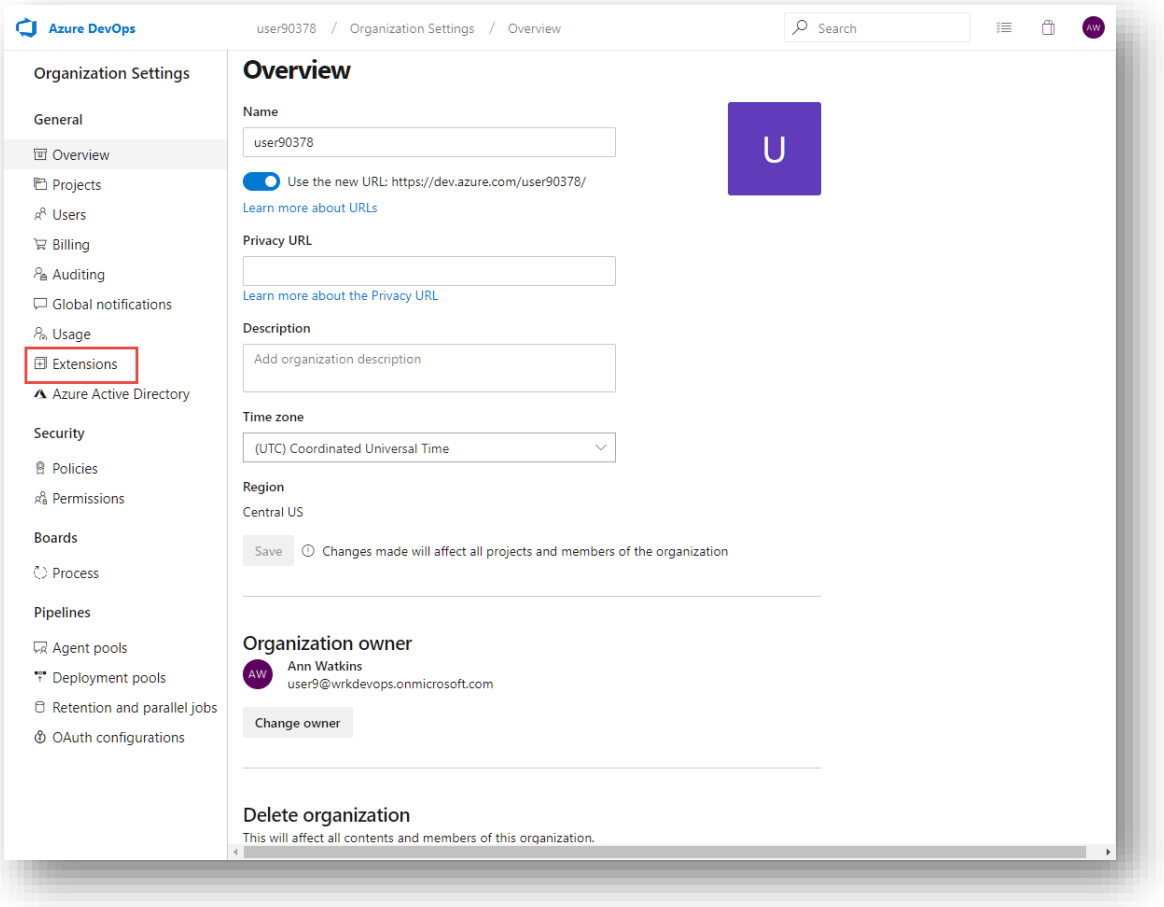


Install and Enable the Azure DevOps Extensions for Power Platform

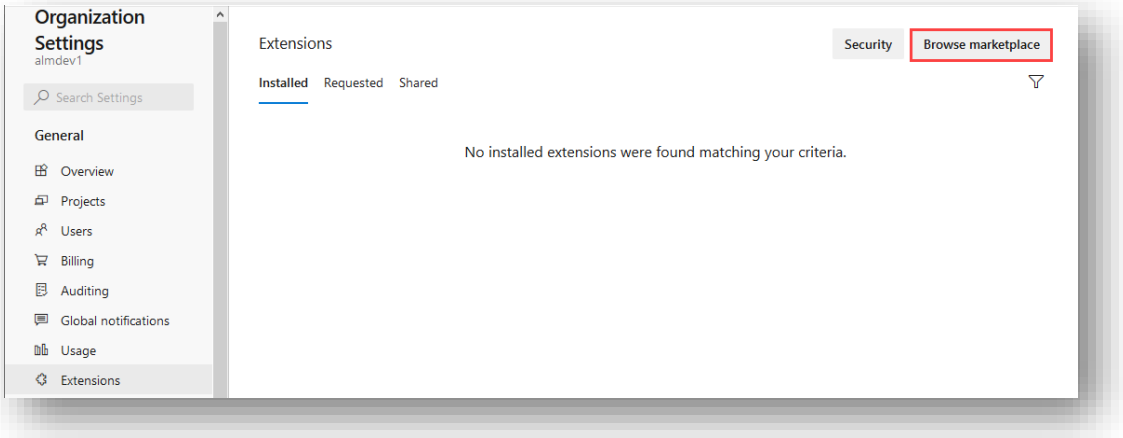
4. On the organization page, click the **Organization settings** link in the bottom left of the navigation panel.



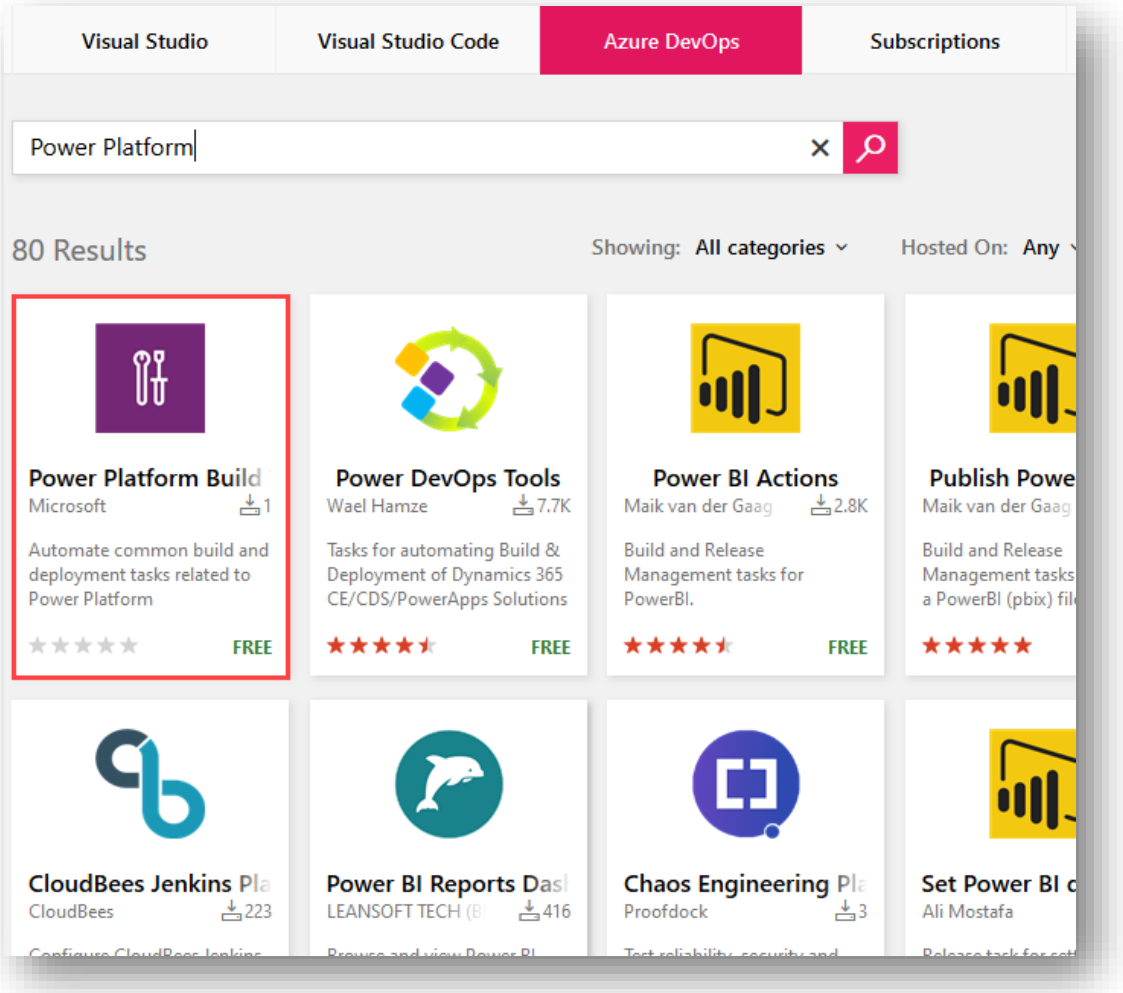
5. On the **Organization Settings** page, click the **Extensions** link in the left navigation panel.



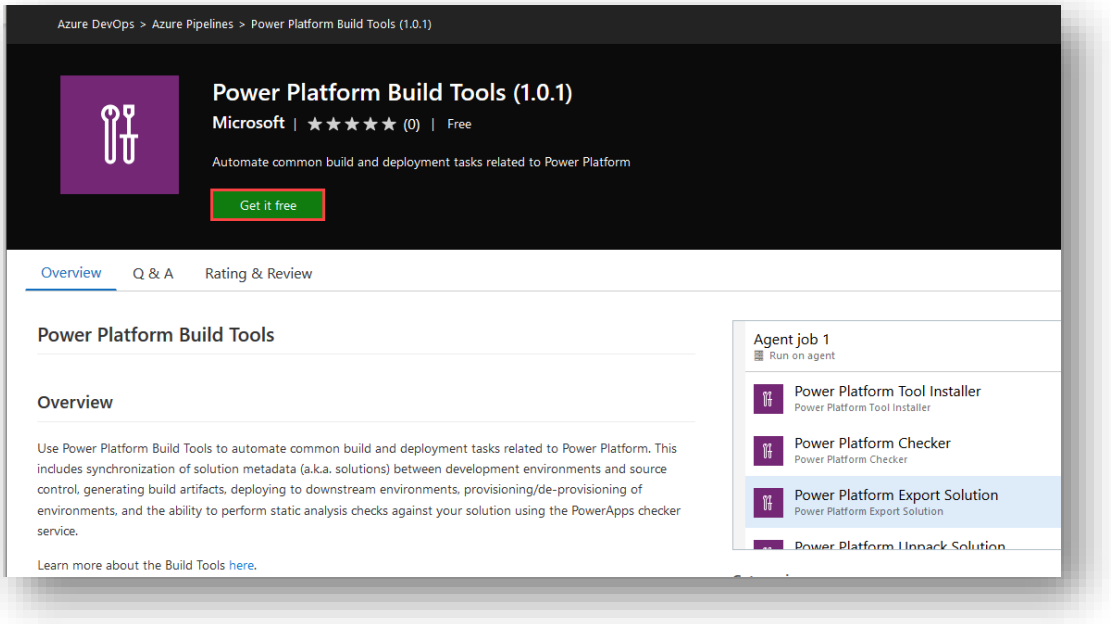
6. Click **Browse marketplace**.



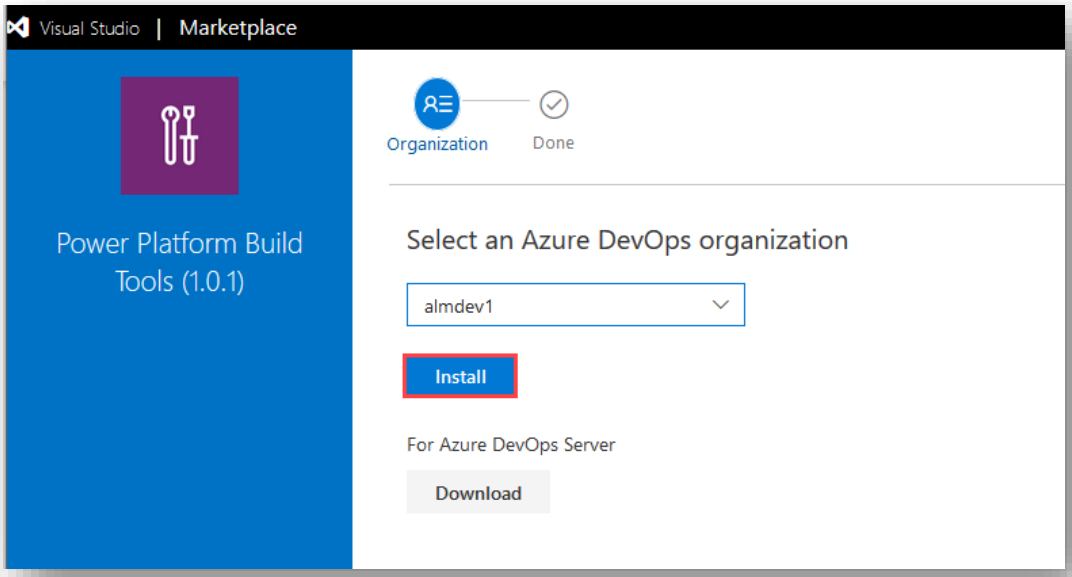
7. This will redirect you to the Azure DevOps marketplace. Search for *Power Platform* and select **Power Platform Build Tools**. Note that there are great community options too.



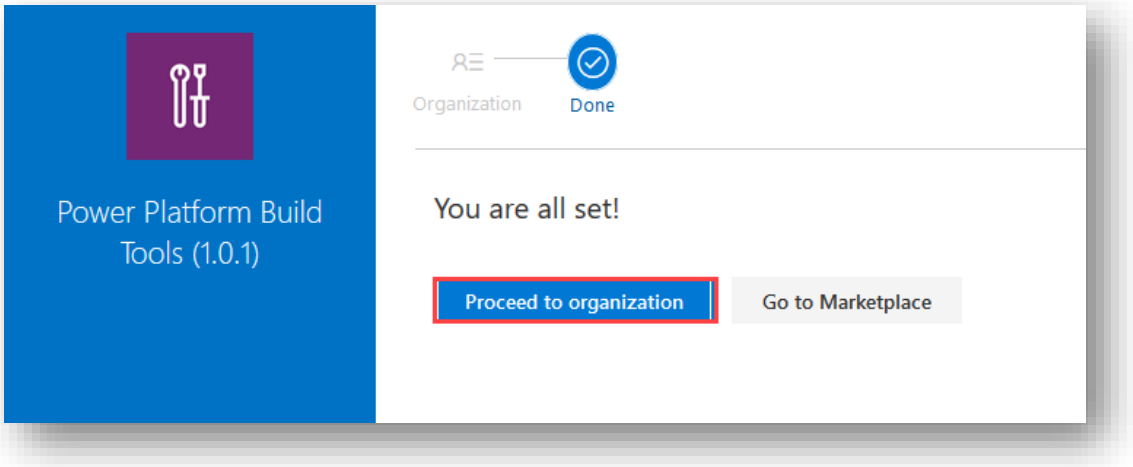
8. Click **Get it free**.



9. Click **Install**.



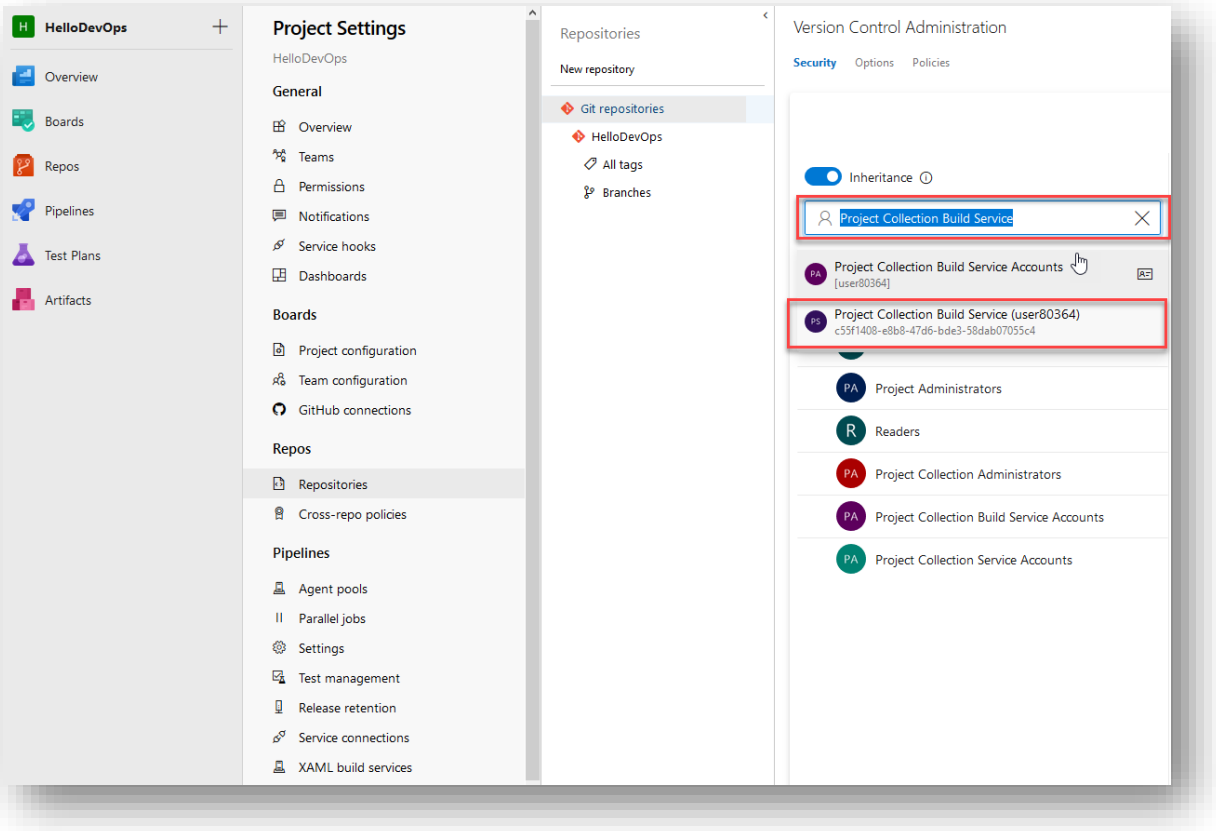
10. Click **Proceed to organization**.



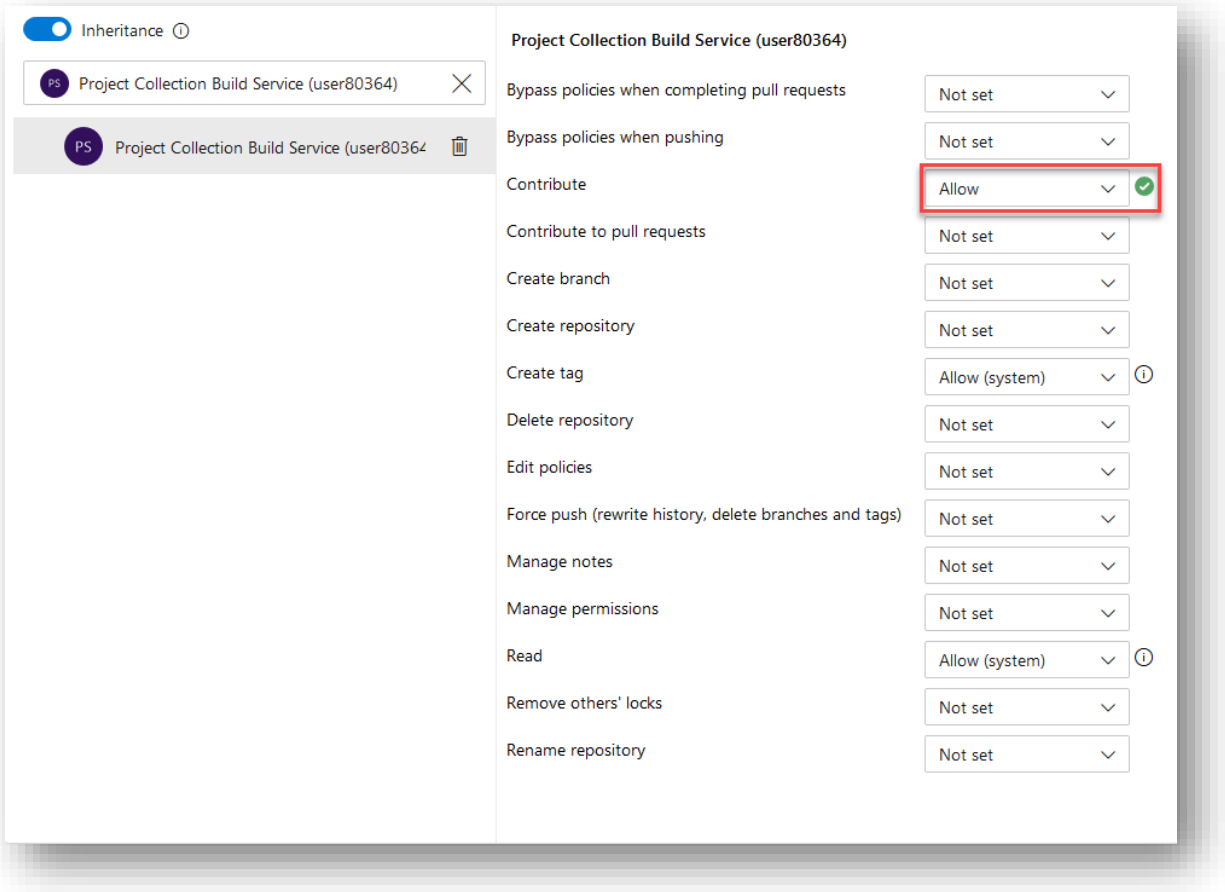
Configure Azure DevOps Permissions for Build Service Account

The build pipelines that we set up later will export files from an organization and check them into your source code repo. This is not a default permission of Azure DevOps, so we need to configure the appropriate permissions for the pipeline to function properly.

11. Click the **Project Settings** icon on the lower left of the screen and click the **Repositories** link in the fly out menu. Type “Project Collection Build Service” in the search box and select it when found. Note: select the project collection with the username appended.



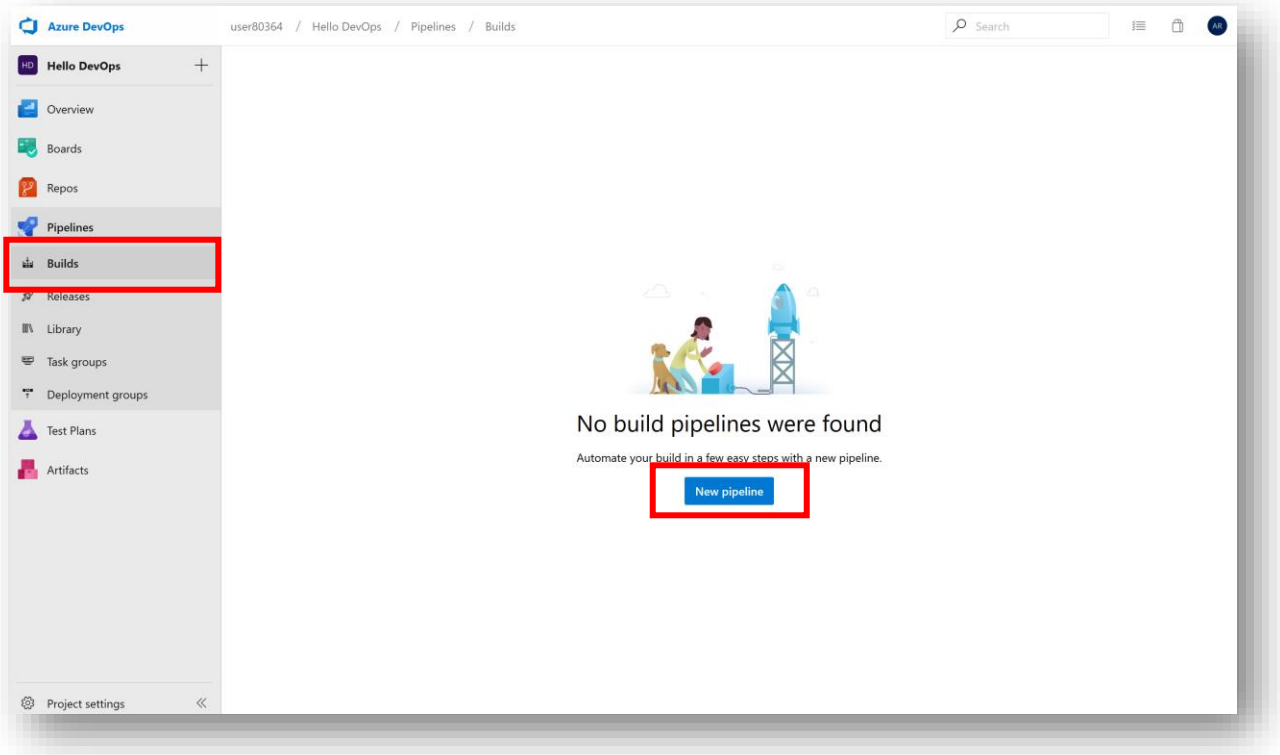
12. The setting for the user is displayed. Select **Allow** for the **Contribute** setting and ensure that the green checkbox is displayed.



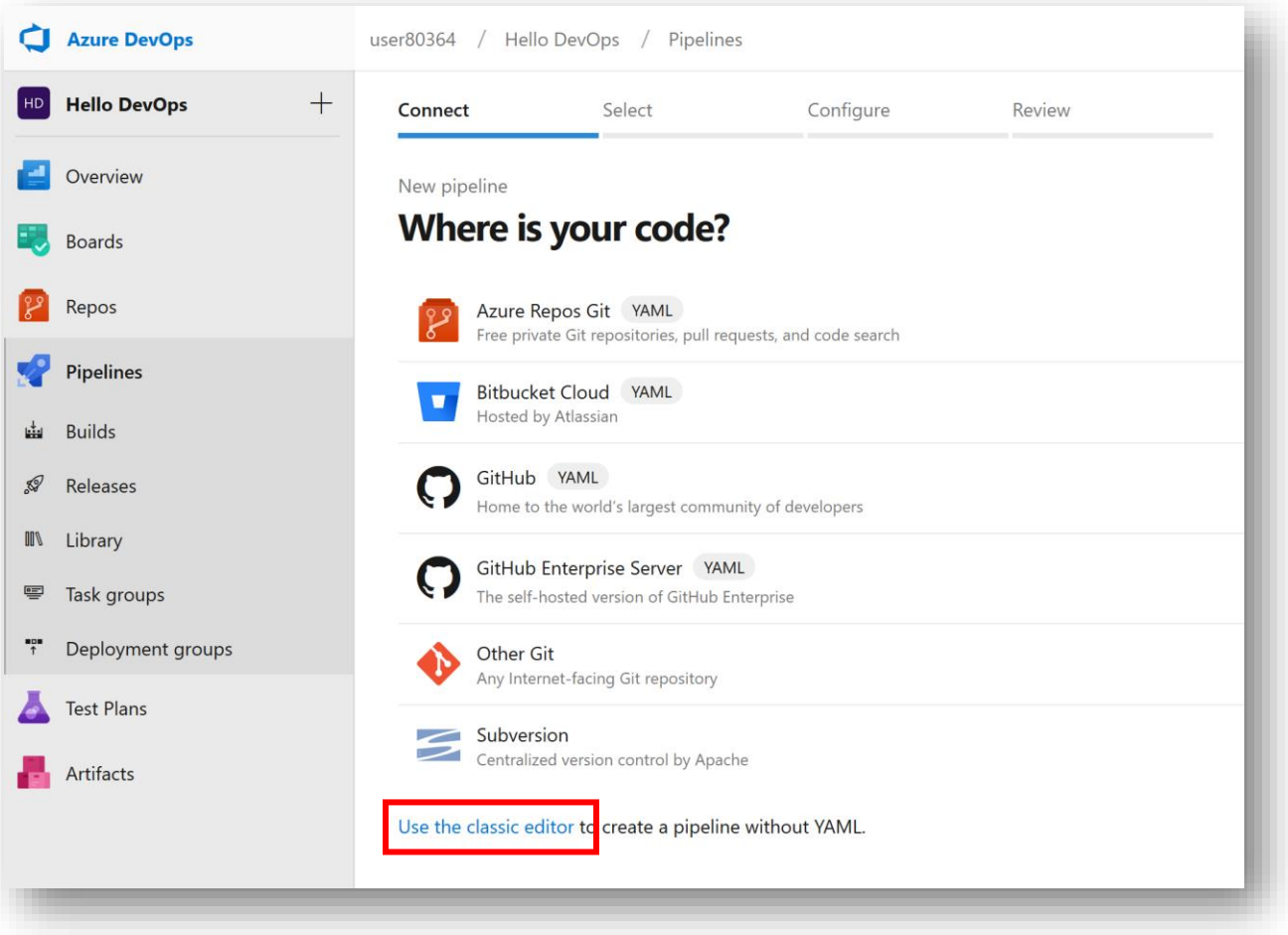
Build Pipeline 1: Create Export from Dev

The first pipeline you will create will export your solution from your development environment as an unmanaged solution, unpack it, and check it into source control (your repo).

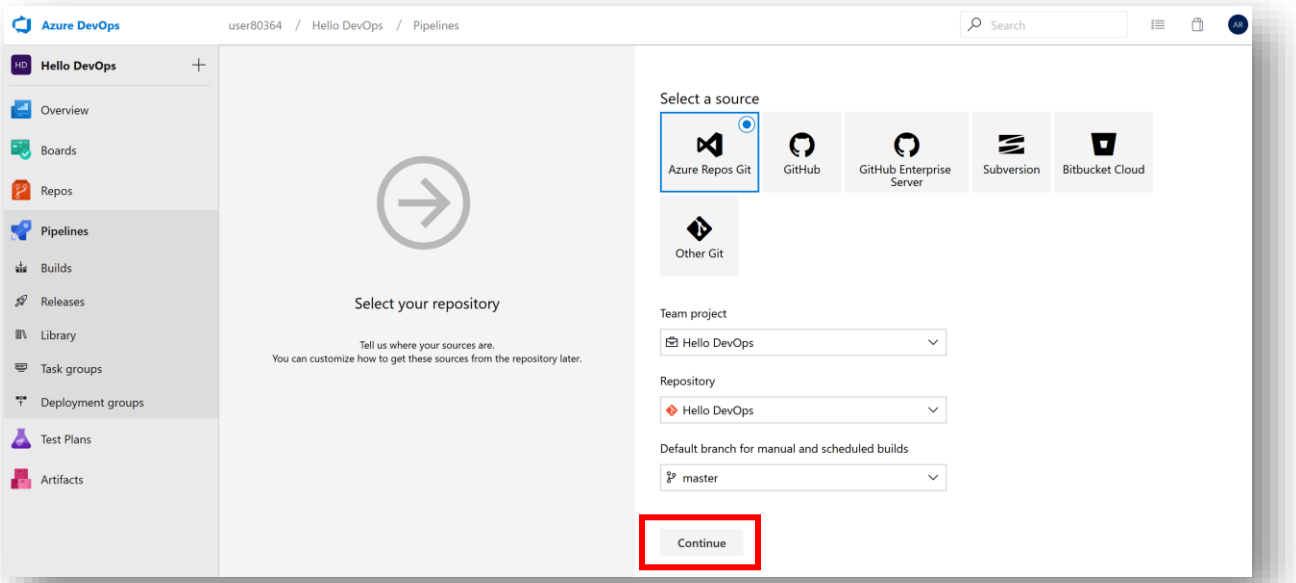
13. Click the **New pipeline** button.



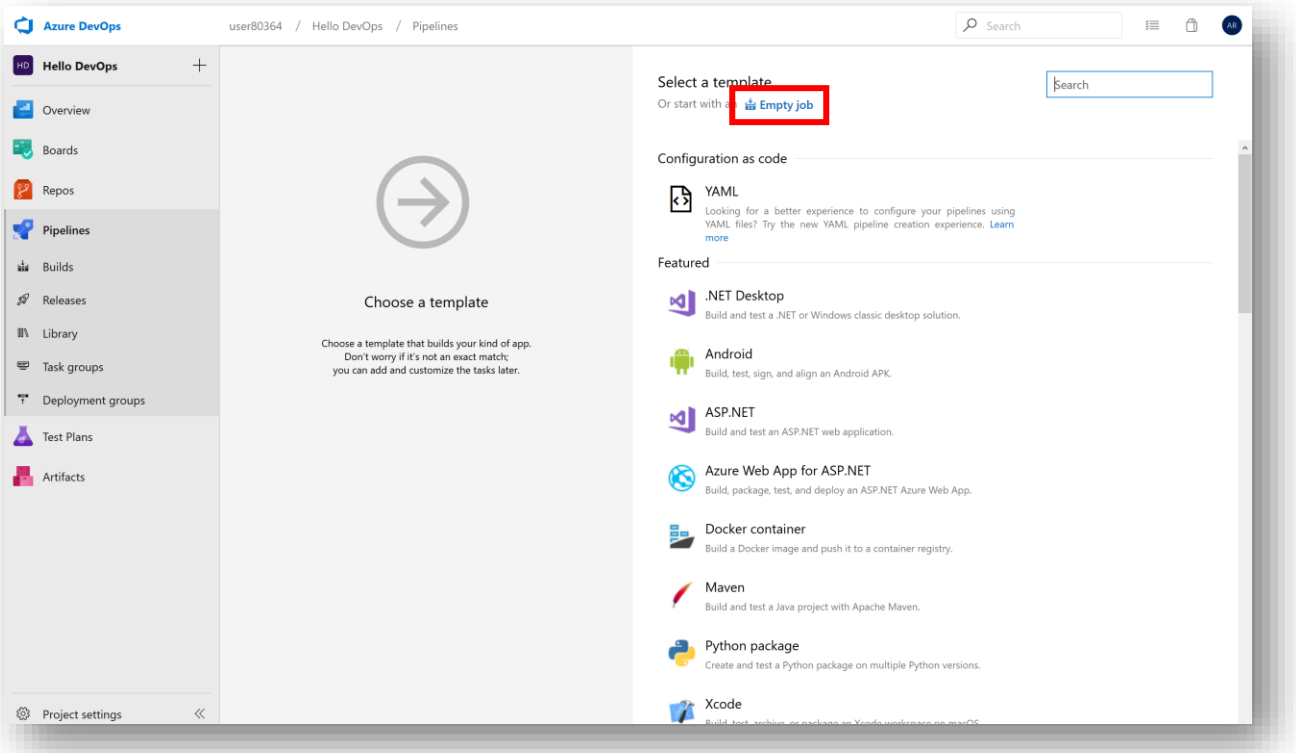
14. Click **Use the classic editor** link to create a pipeline.



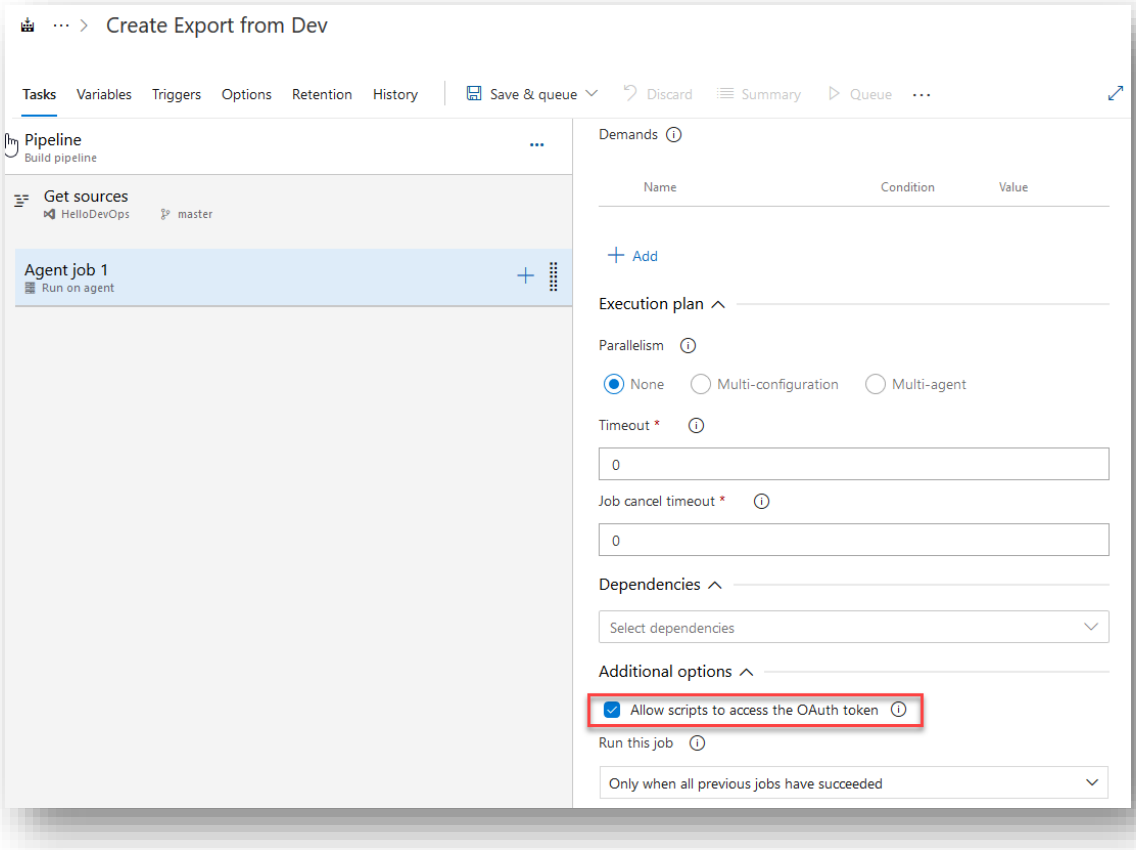
15. Leave default values as is and click **Continue**.



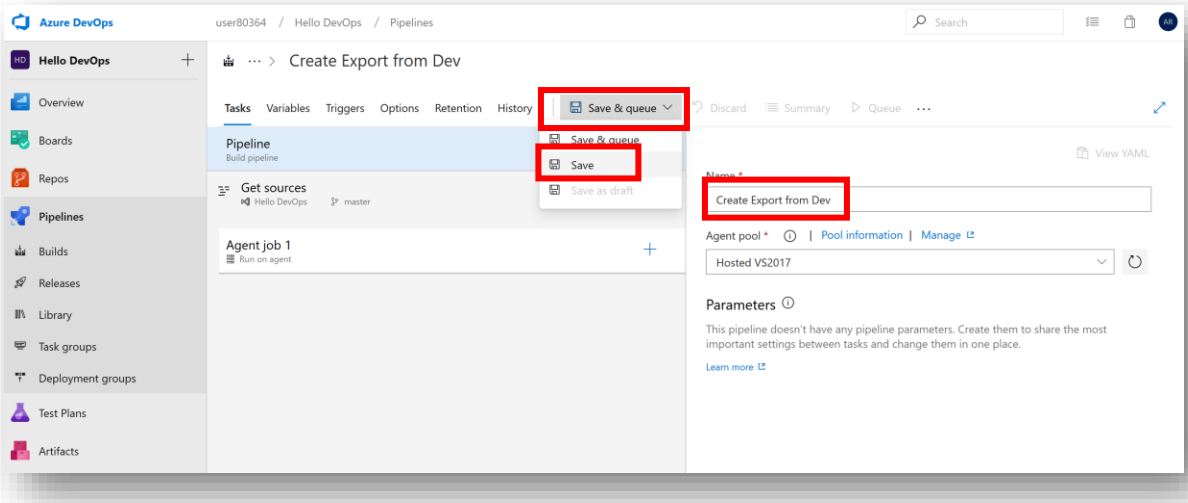
16. Select **Empty job** to create an empty job.



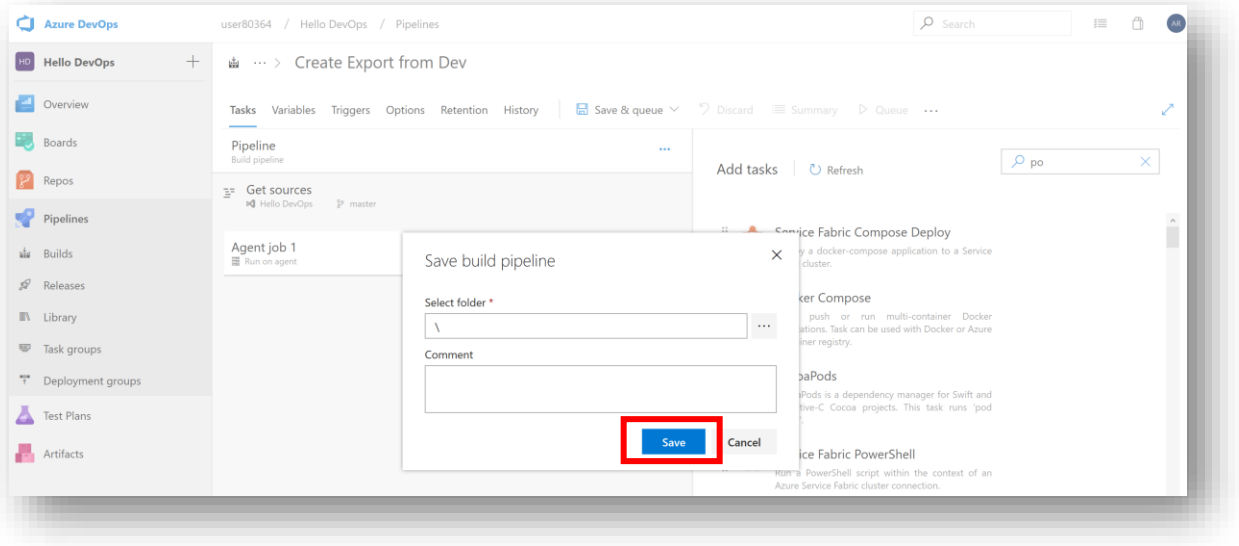
17. Select **Agent job 1** and enable the **Allow scripts to access the OAuth token** option.



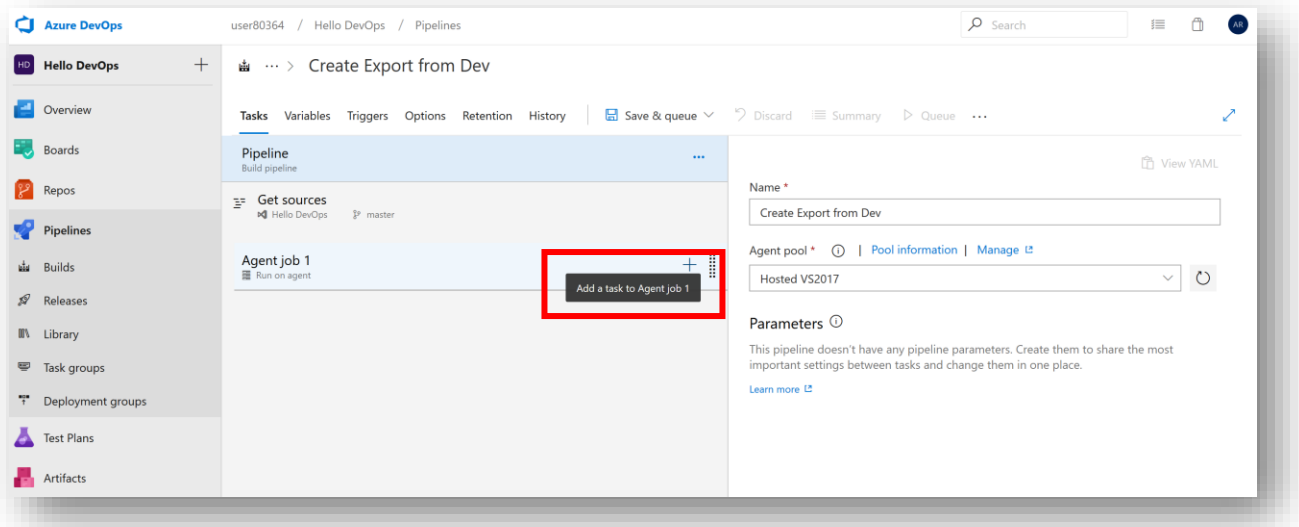
18. Name the pipeline “Create Export from Dev” and click **Save**.



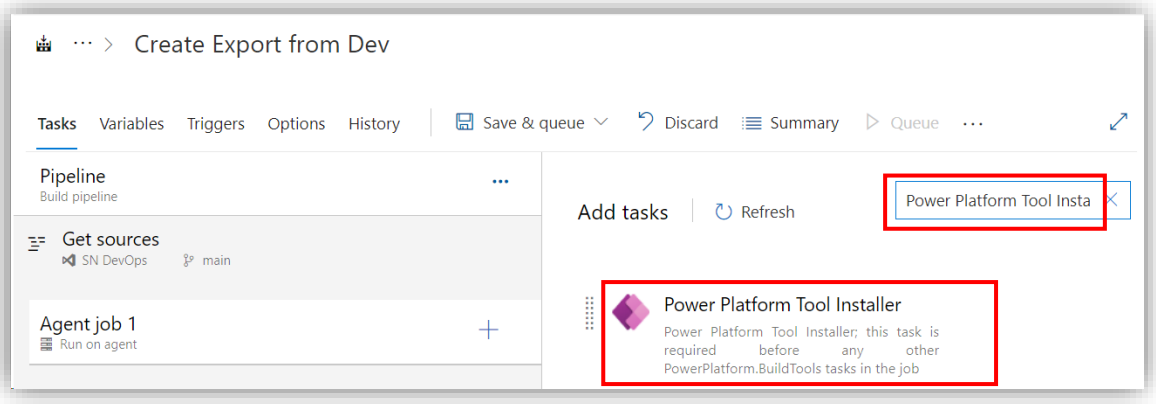
19. Changes to your pipeline are also checked into source control, so select the folder, type a comment, and click **Save**.



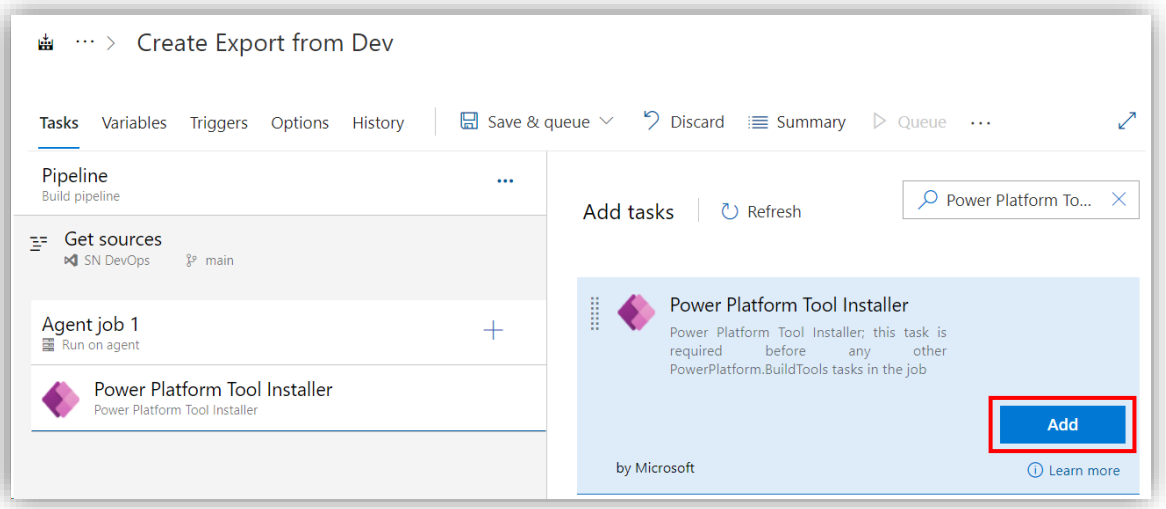
20. You are now ready to start using the Power Platform Build Tools tasks for Azure DevOps in your pipeline. Select **Add a task to Agent job 1**.



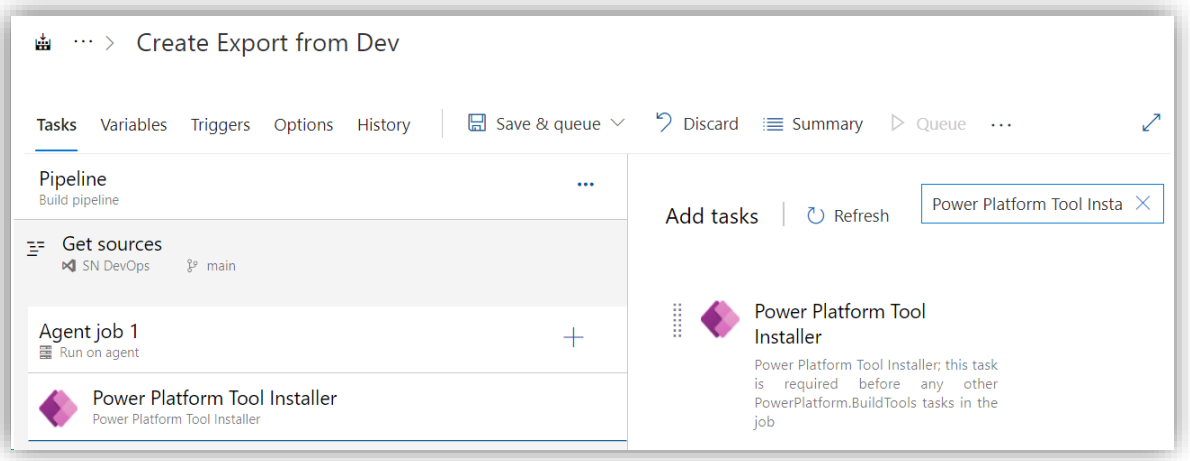
21. The very first task you will perform is one that installs the required tools on the agent. Search for "Power Platform Tool".



22. Select **Add** under the **Power Platform Tool Installer** task.



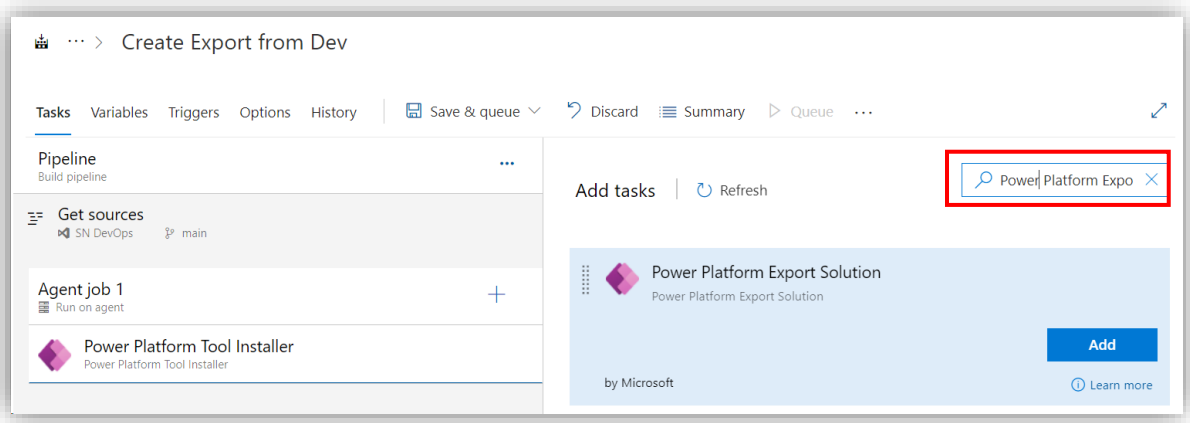
23. This will add the tool Installer task to the pipeline. No additional configuration is required for this task.



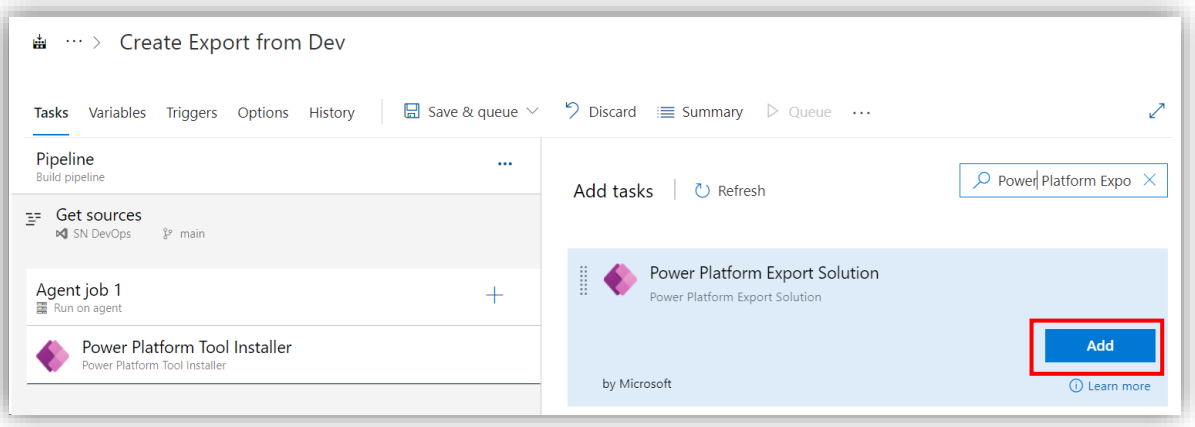
Export Solution as Unmanaged

Your unmanaged solution is your source code for your configuration. You should export your solution as unmanaged to check it into source control.

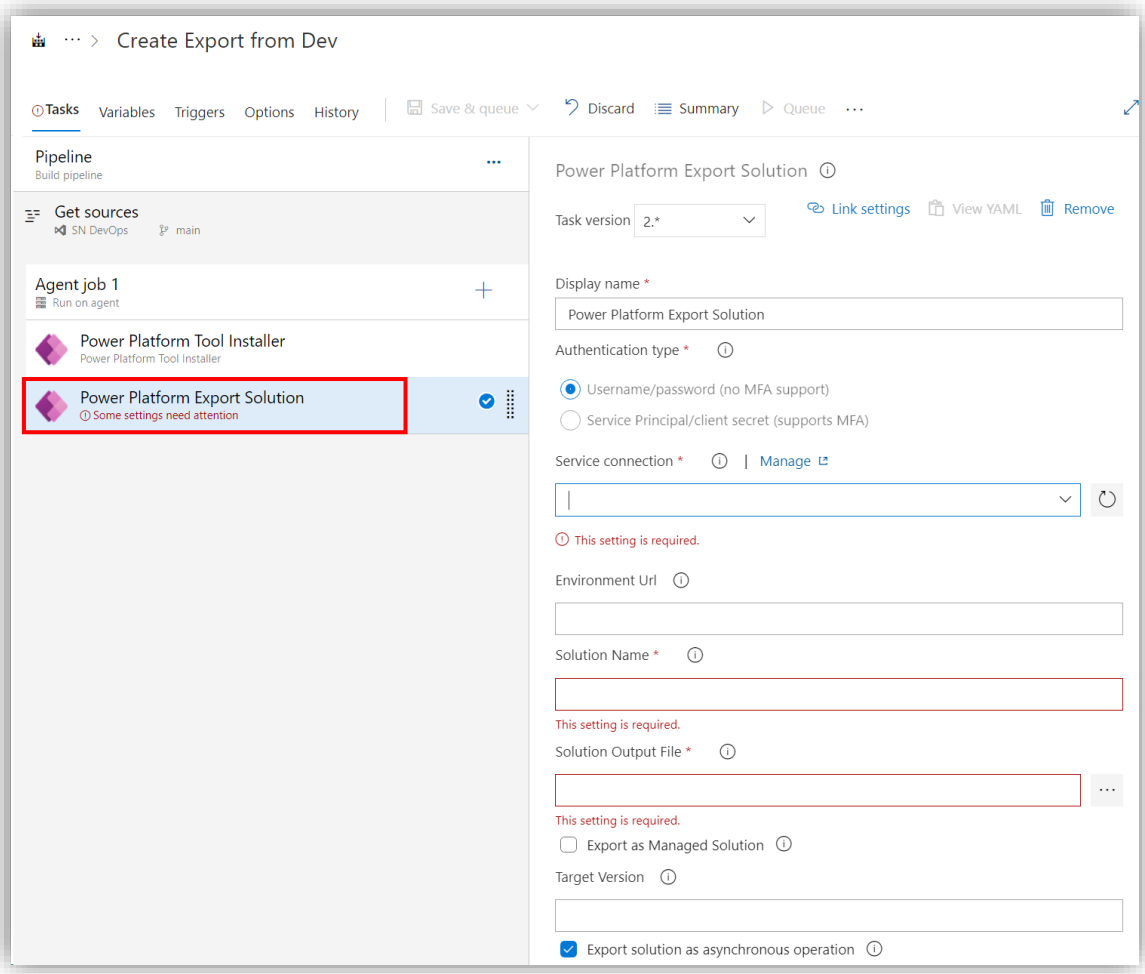
24. The next task you will add is a task that exports the unmanaged solution from development environment. Search for “Power Platform Export Solution”.



25. Select the task and Click **Add**.



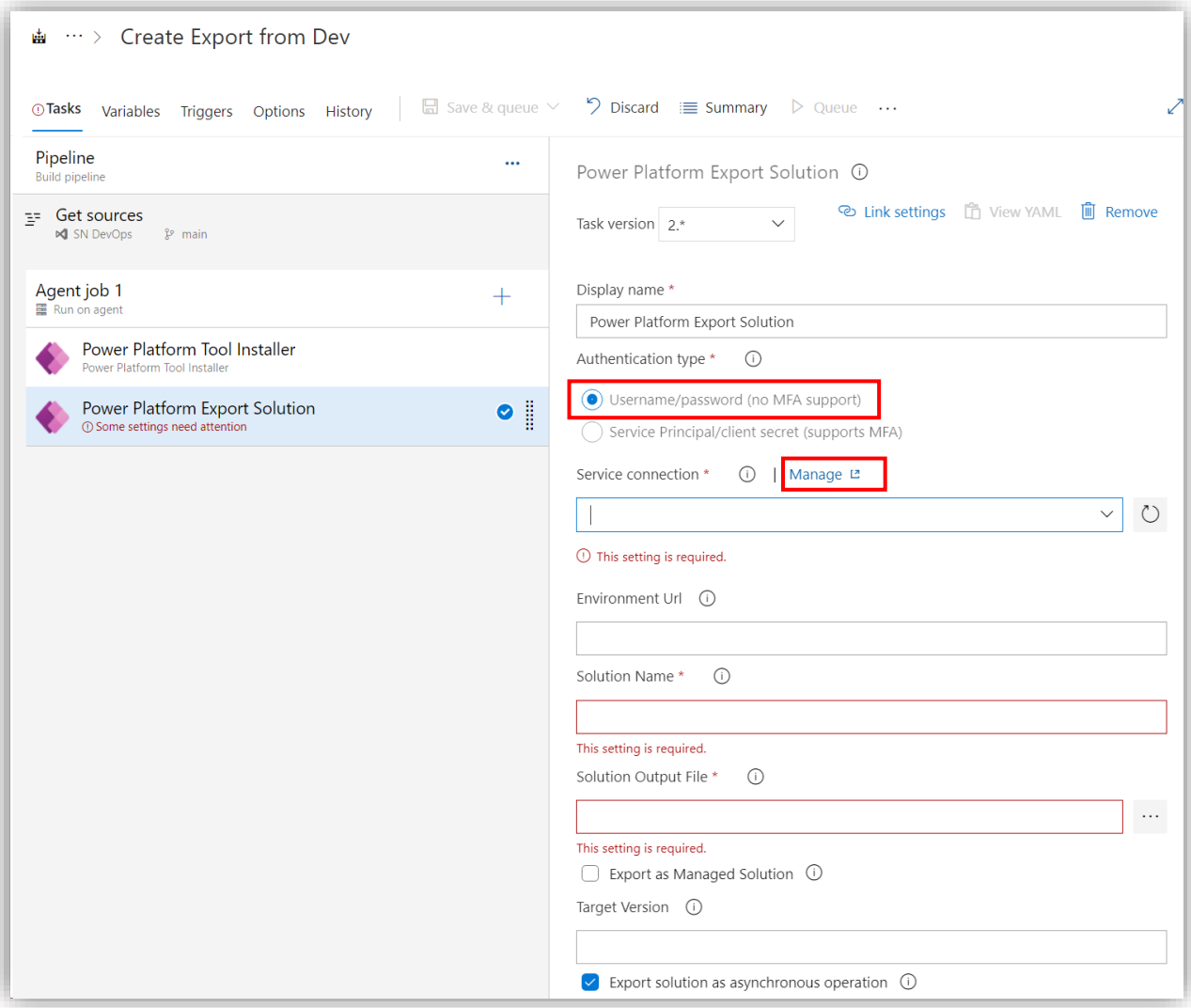
26. After adding the Power Platform Export Solution task, you will notice that additional configuration is required. Click on **Power Platform Export Solution** in the pipeline view.



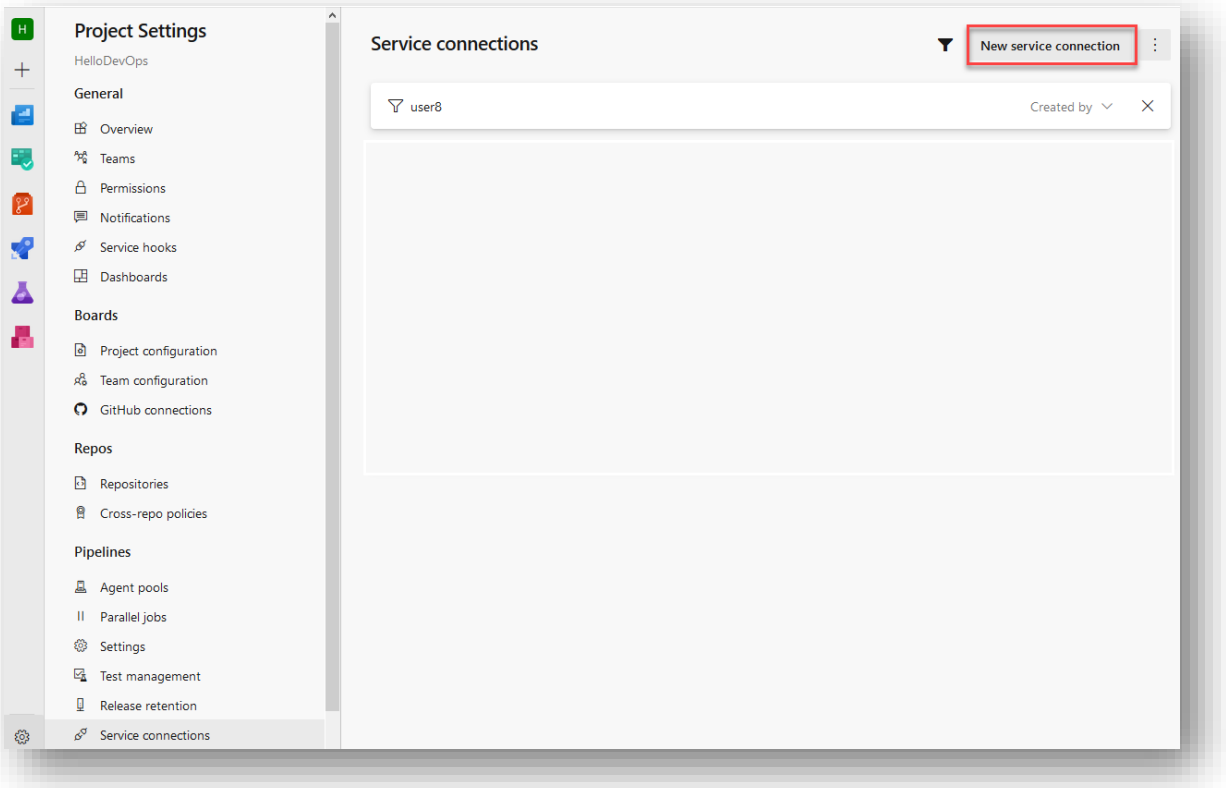
27. This will open the task configuration page:

- **Display name** is inherited from build task itself.
- **Authentication type**: Two types of authentications are available:
    - **Username/password**: simple to setup but does not support multi-factor authentication (MFA). This is what we will use for this lab.
    - **Service Principal/client secret**: recommended and supports MFA. This is harder to setup as it requires creation of the Service Principal and client secret in the Azure Portal as well as creation of the application user in the Power Platform environment. Not used in this lab but anyone familiar with setting this up can use this option instead throughout the rest of the lab.
- **Service Connection**: This is the connection to the environment that you want to export the solution from. We will define this in the next step.
- **Solution Name**: This is the name of the solution you want to export.
- **Solution Output File**: This specifies the path and filename of the generated solution compressed file (zip).
- **Export as managed solution**: By default, solutions are exported as unmanaged (for development purposes). Setting this flag exports the solution as Managed (used for deployment to any downstream environment such as Test, Pre-Prod and Production).

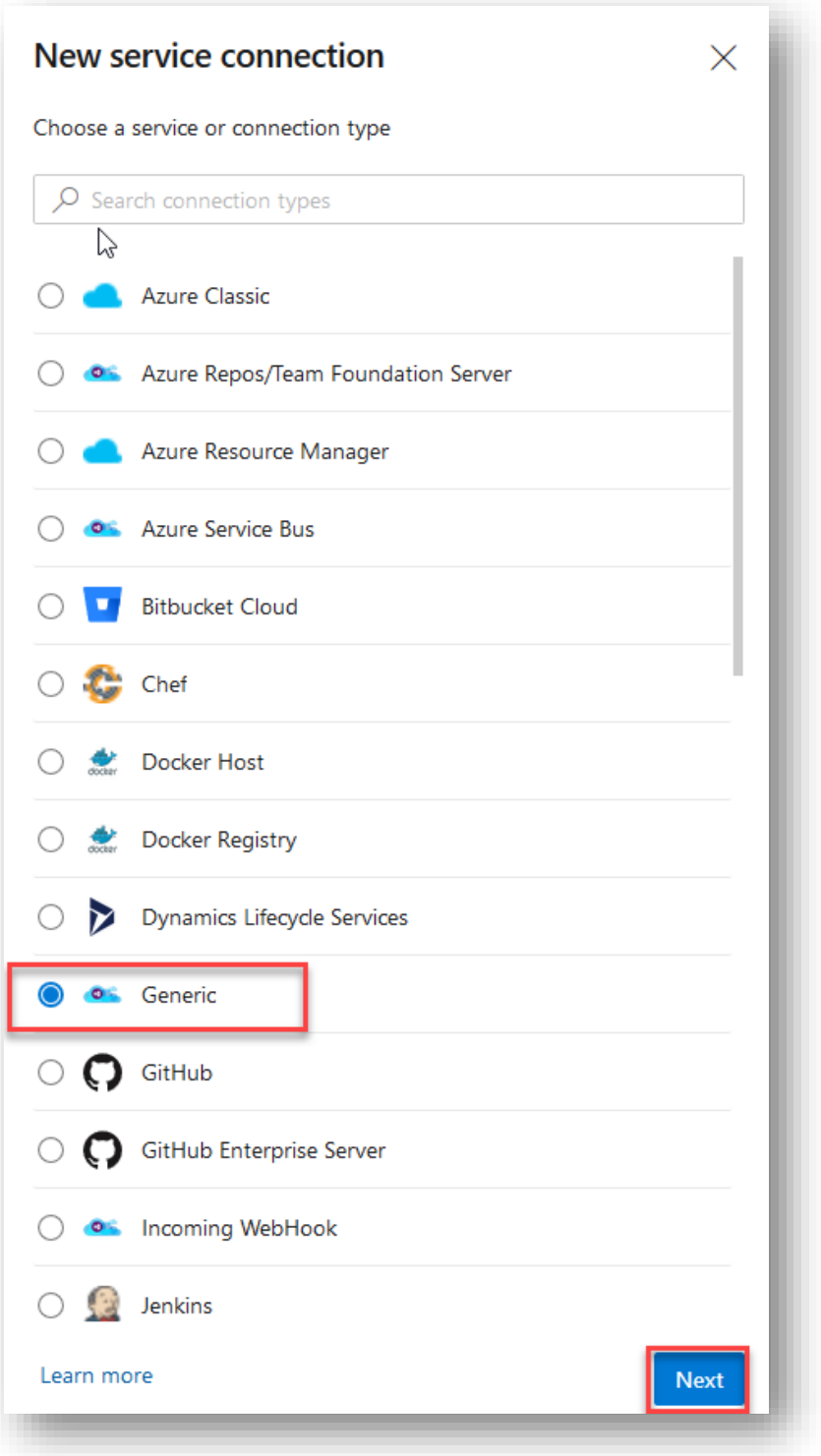
28. Select **Username/password** under **Authentication type**. Click on **Manage** next to **Service connection**.



29. This will bring up the connection configuration required to export the solution from the development environment **(user-xx-dev).** Click **New service connection**.

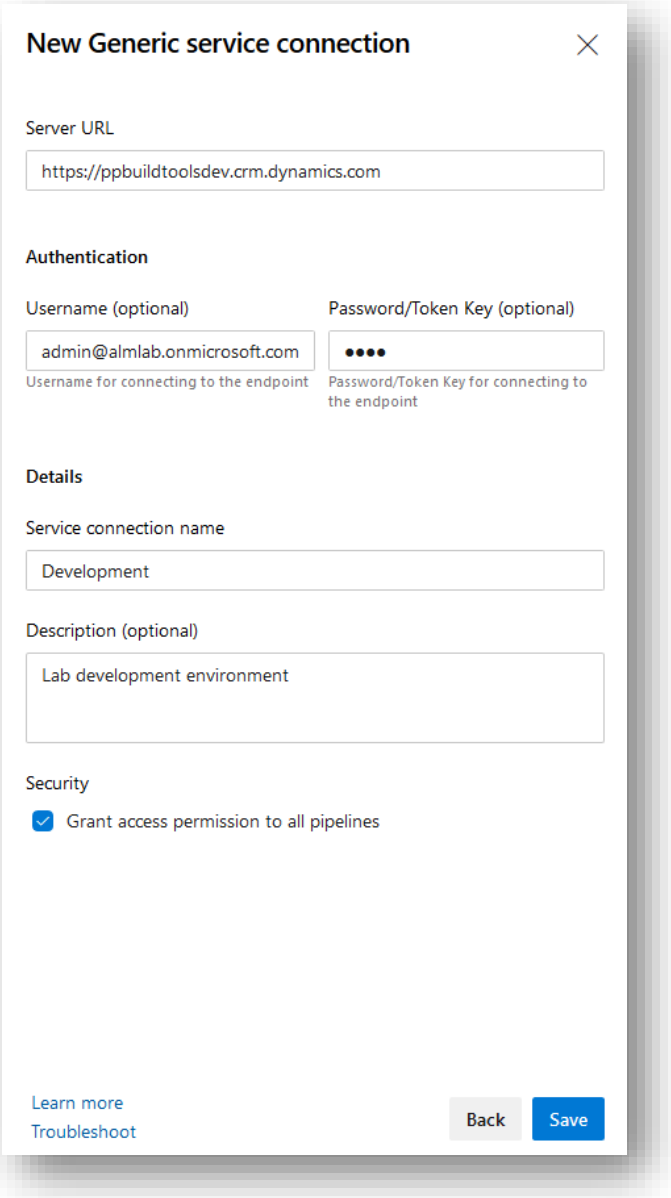


30. This will bring up the **New service connection** screen. Select **Generic** from the list of connections and click **Next**.

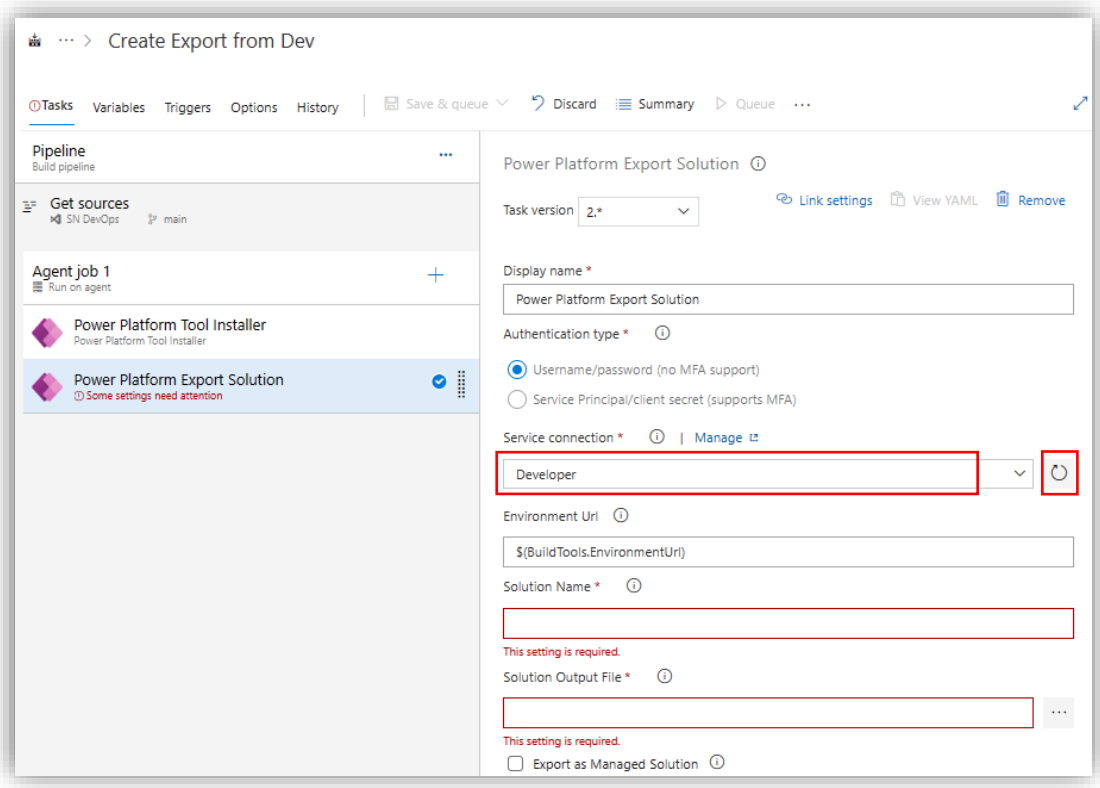


31. Fill out the required details:
    - **Server URL**: https://<environment-url>.crm.dynamics.com. This should be the URL to your development environment.
    - **Username:** username of a user with administrator access to the environment.
    - **Password:** Associated password for the user.
    - **Service connection name:** This name will be used to identify which environment to connect to in the Build tasks.
    - **Description:** Give the connection a description that helps you identify the environment the service connection connects to.

    A sample connection is shown below.

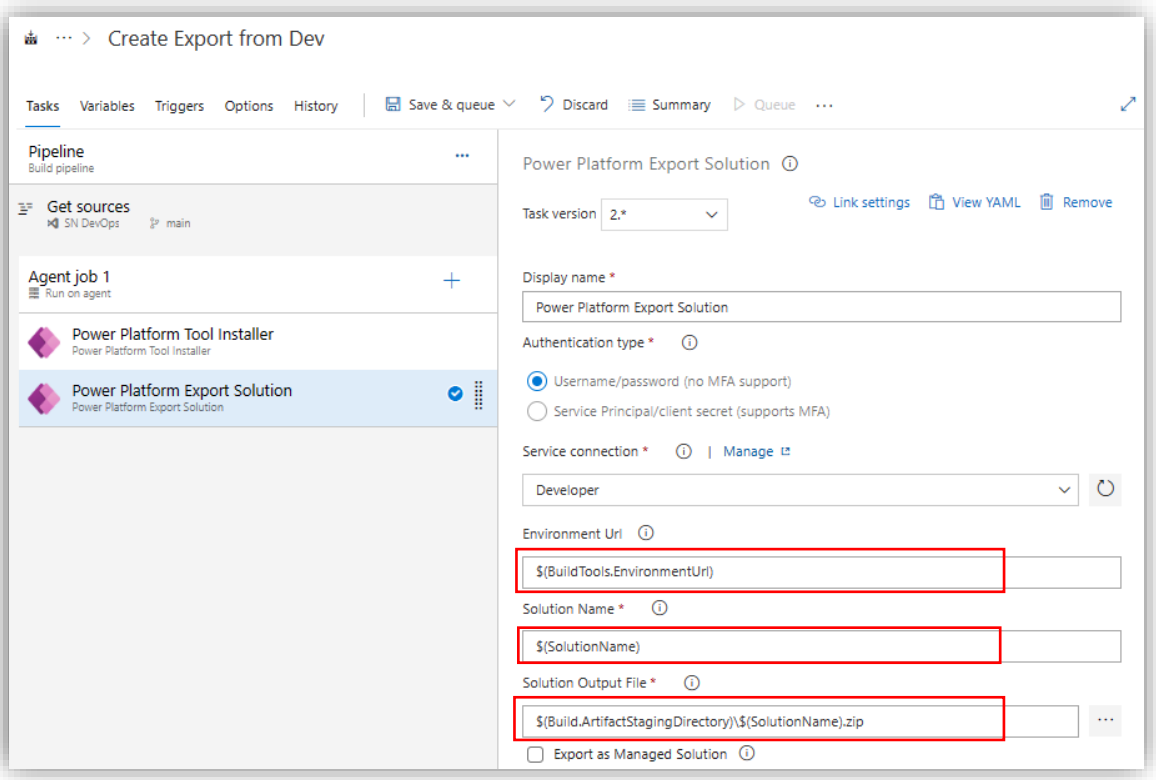


32. Click **Save**. You will be in the pipeline service connections area in your project.
33. Close the browser tab and go back to the previous tab where you were building the pipeline. Select the service connection that was created in the previous step.
    Note: You might have to click the refresh button before the newly created service connection is available to select.



34. Fill out the remaining details:

- Solution Name: **$(SolutionName)**
  Note: This will use the input parameter that you specify when running (queuing) the build pipeline.
- Solution Output File: **$(Build.ArtifactStagingDirectory)\$(SolutionName).zip**
  Note: This will add the file to your repo and retain the existing solution name.

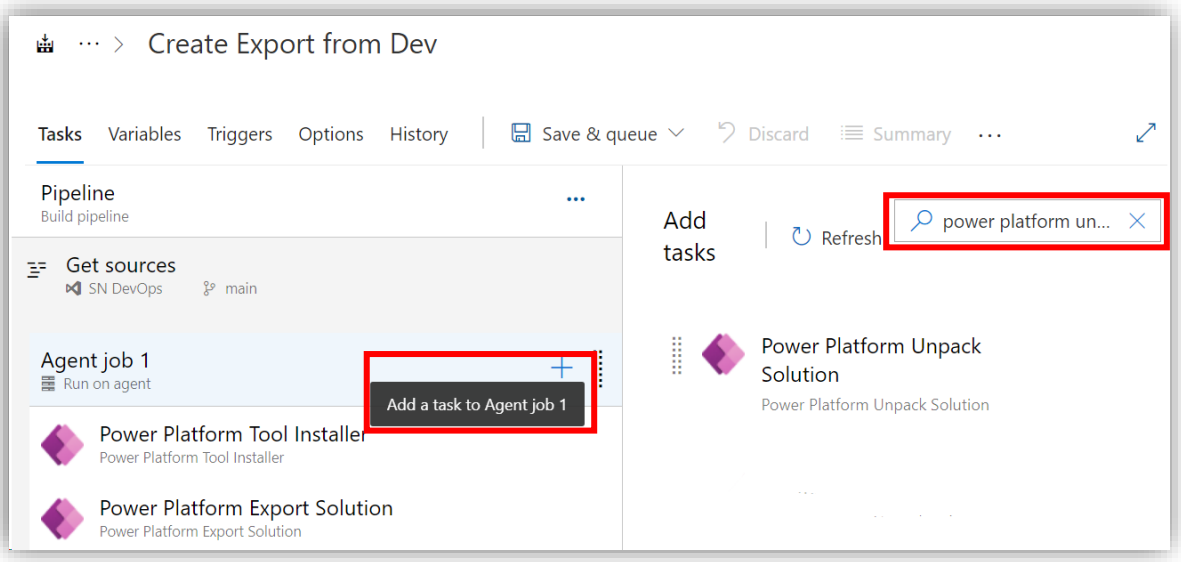


35. Save the build pipeline by clicking **Save & queue**, then save from the top command bar and clicking **Save** on the dialog.

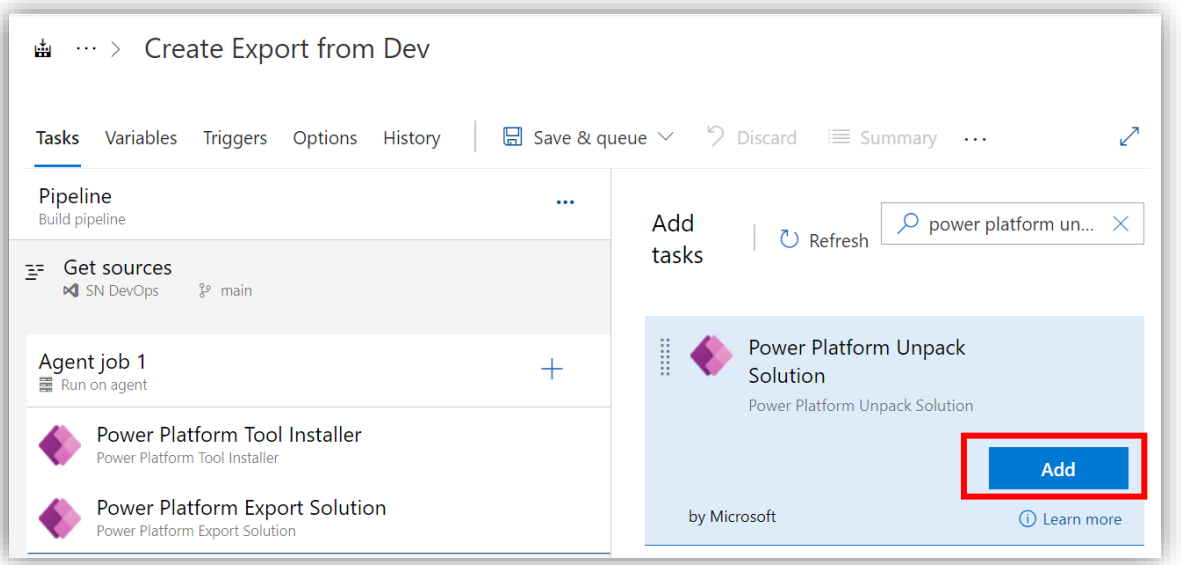
Unpack Solution

The solution file that is exported from the server is a compressed zip file with consolidated configuration files. These initial files are not suitable for source code management as they are not structured to make it feasible for source code management systems to properly do differencing on the files and capture the changes you want to commit to source control. You need to 'unpack' the solution files to make them suitable for source control storage and processing.

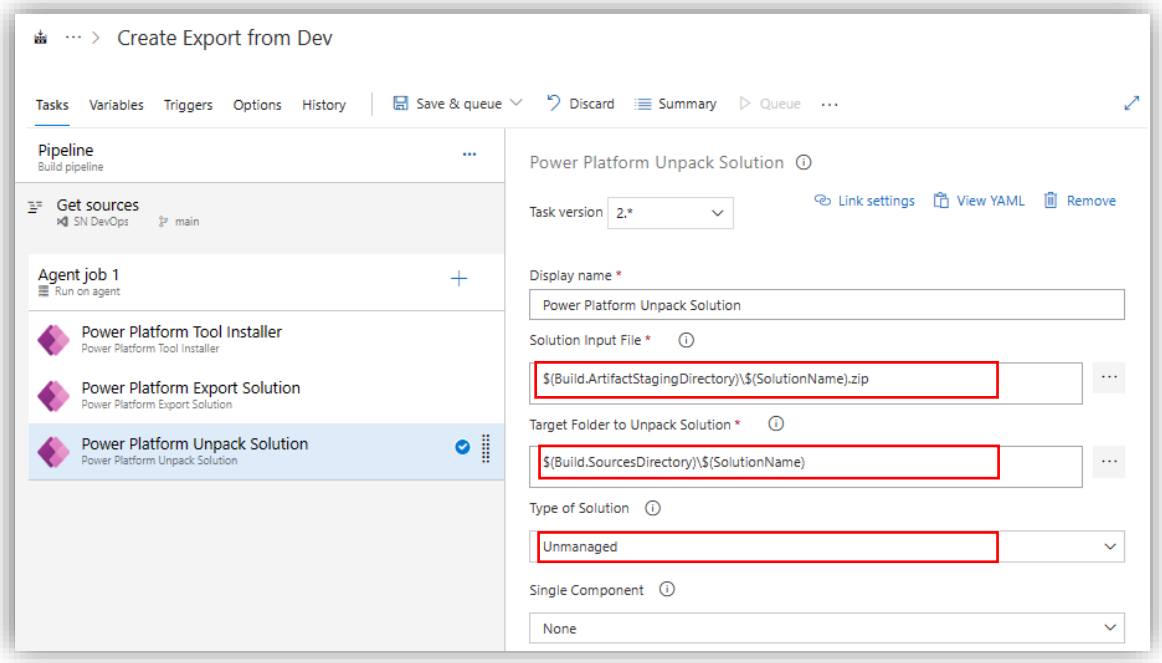
36. Click **Add a task**, then search for **Power Platform Unpack**.



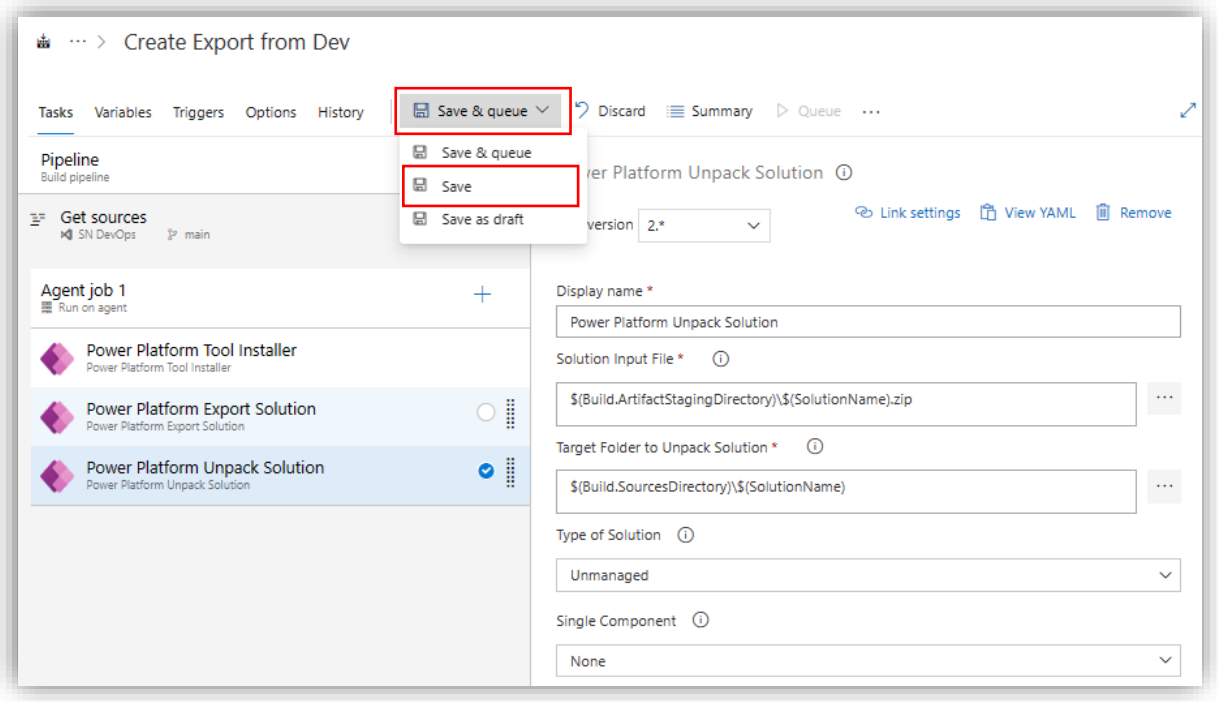
37. Add the **Power Platform Unpack Solution** task to the pipeline.



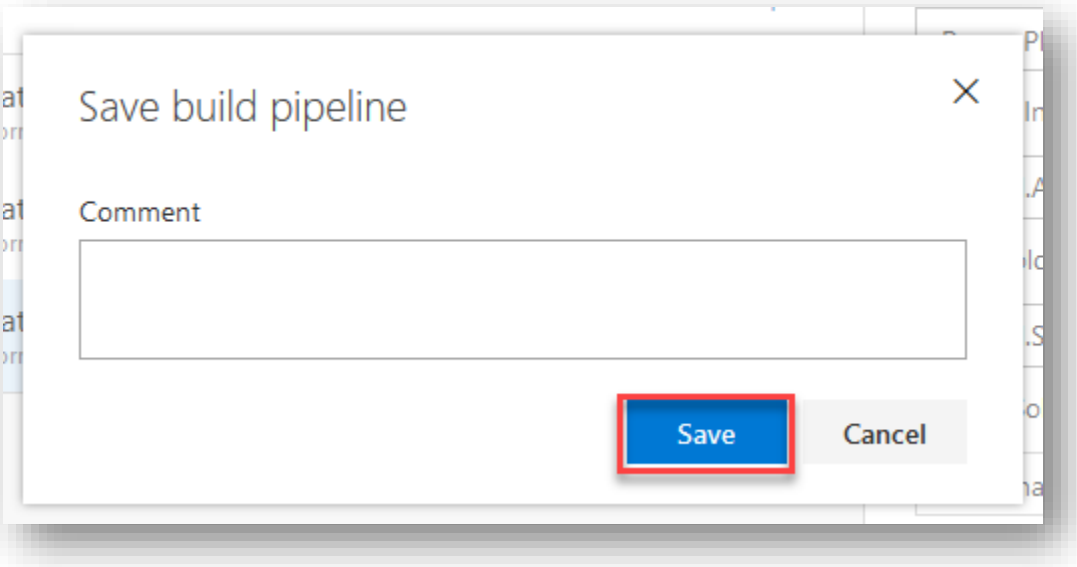
38. Open the task to configure settings for the task with the following details:
    - Solution Input File: **$(Build.ArtifactStagingDirectory)\$(SolutionName).zip**
    - Target Folder to Unpack Solution: **$(Build.SourcesDirectory)\$(SolutionName)**
    - Type of solution: **Unmanaged**



39. Save the updated pipeline.



40. You can leave the comment blank.

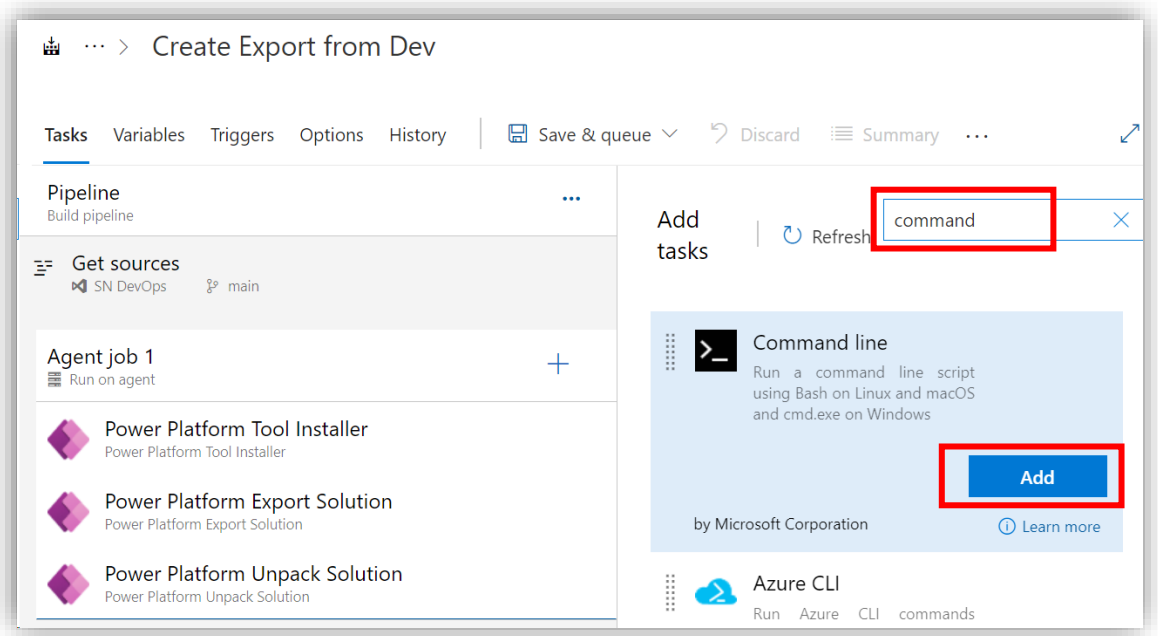


Commit Solution Files to Source Control

Next, we are going to add some scripts as a next pipeline step to commit the solution to the repo. The reason we allowed scripts to access the OAuth token when we started building the pipeline was to allow

for this next step.

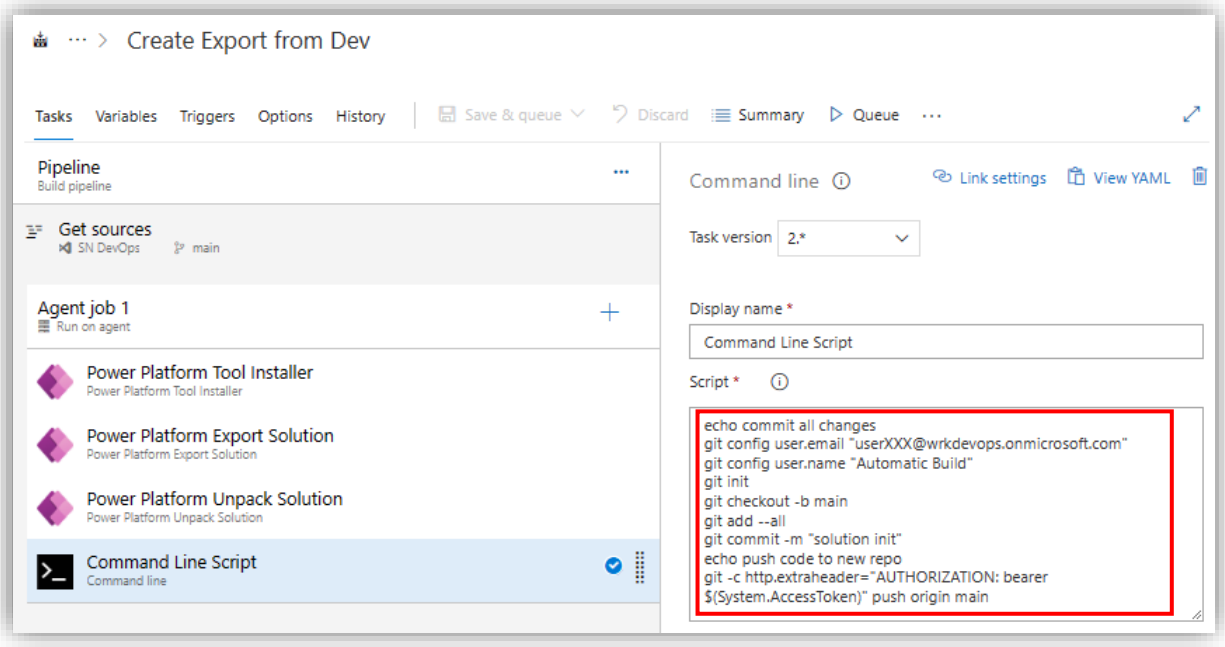
41. Add a Command Line task to your pipeline by clicking the **+** button, searching for “command”, and adding it.



42. Select the task, give the task a display name and copy the following script into the script textbox:

```
echo commit all changes
git config user.email "userXXX@wrkdevops.onmicrosoft.com"
git config user.name "Automatic Build"
git init
git checkout -b main
git add --all
git commit -m "solution init"
echo push code to new repo
git  -c http.extraheader="AUTHORIZATION: bearer $(System.AccessToken)" push origin main
```

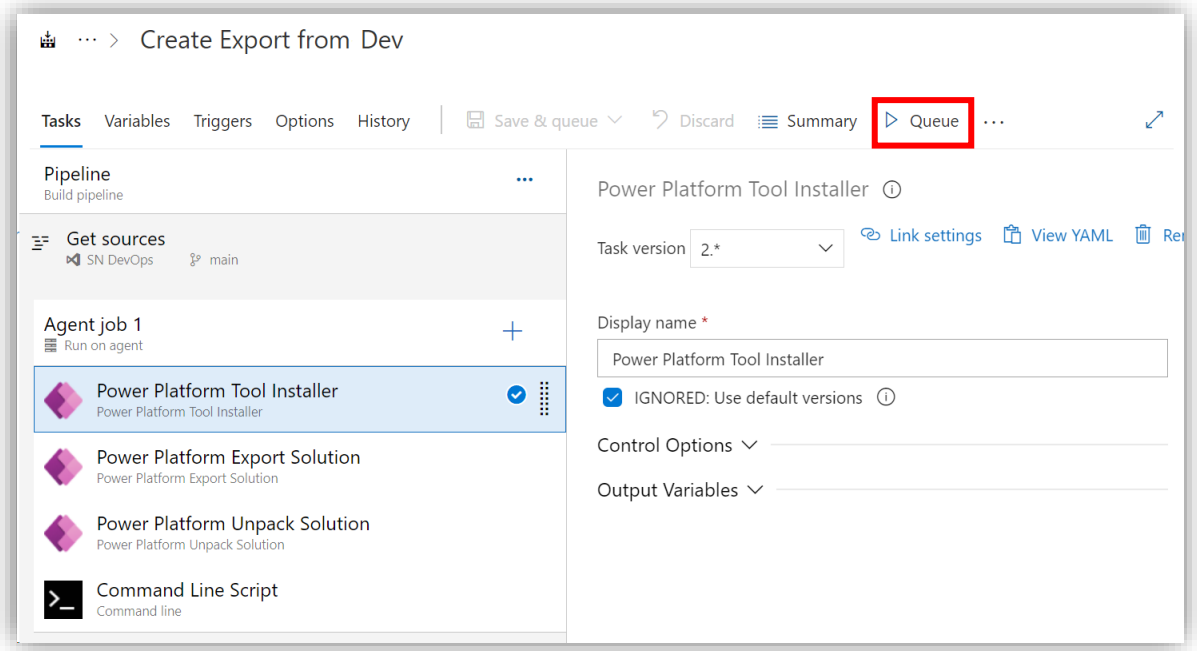
43. Screenshot provided below:



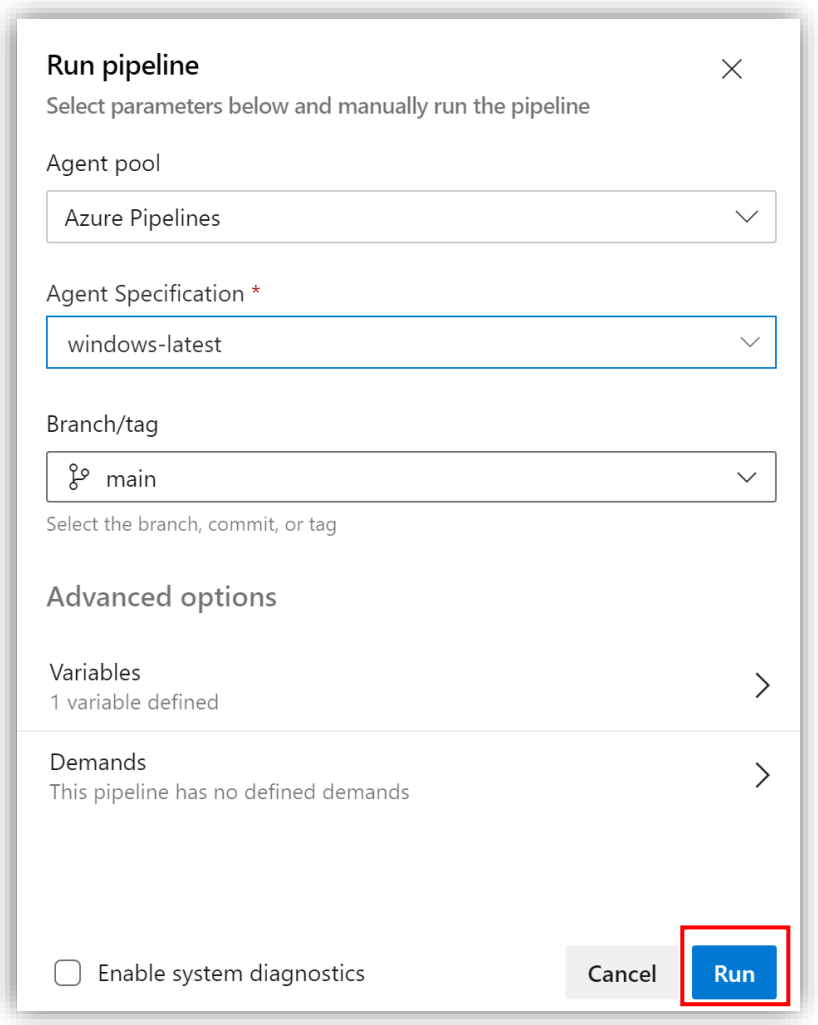
44. Save the task (**Save & queue** > **Save**).

Test your Pipeline

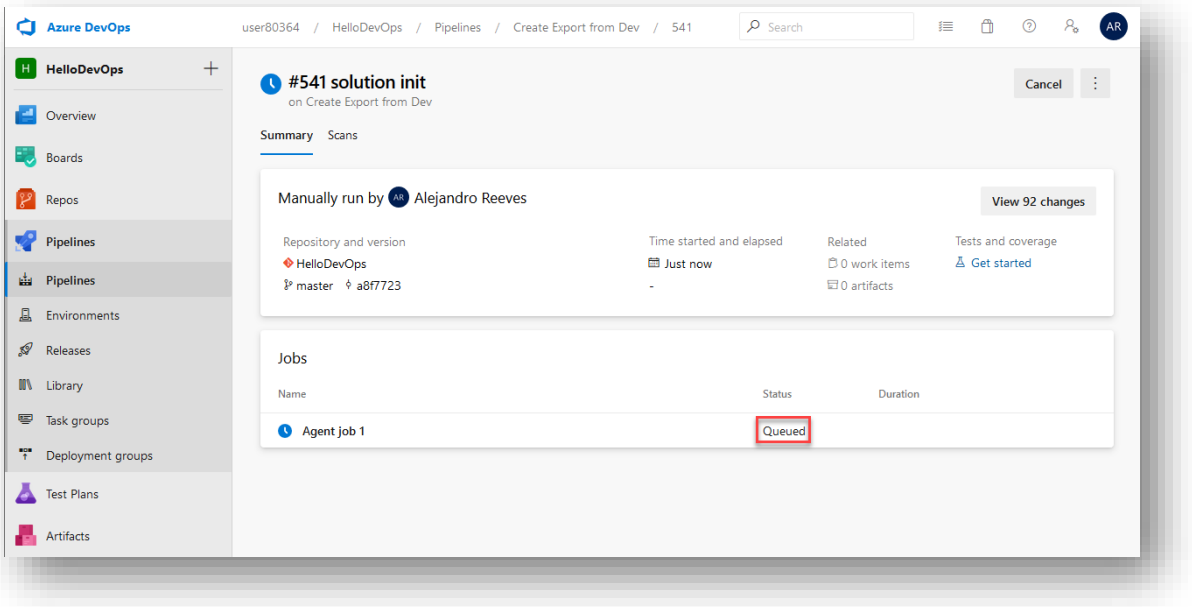
45. Select **Queue** to execute your pipeline.



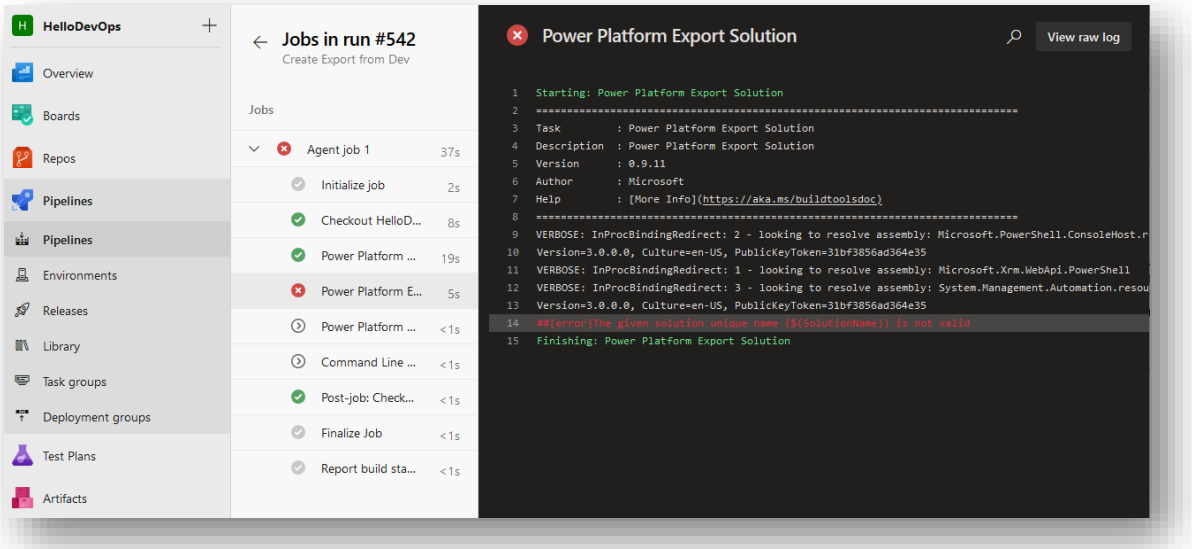
46. Leave the defaults and click **Run**.



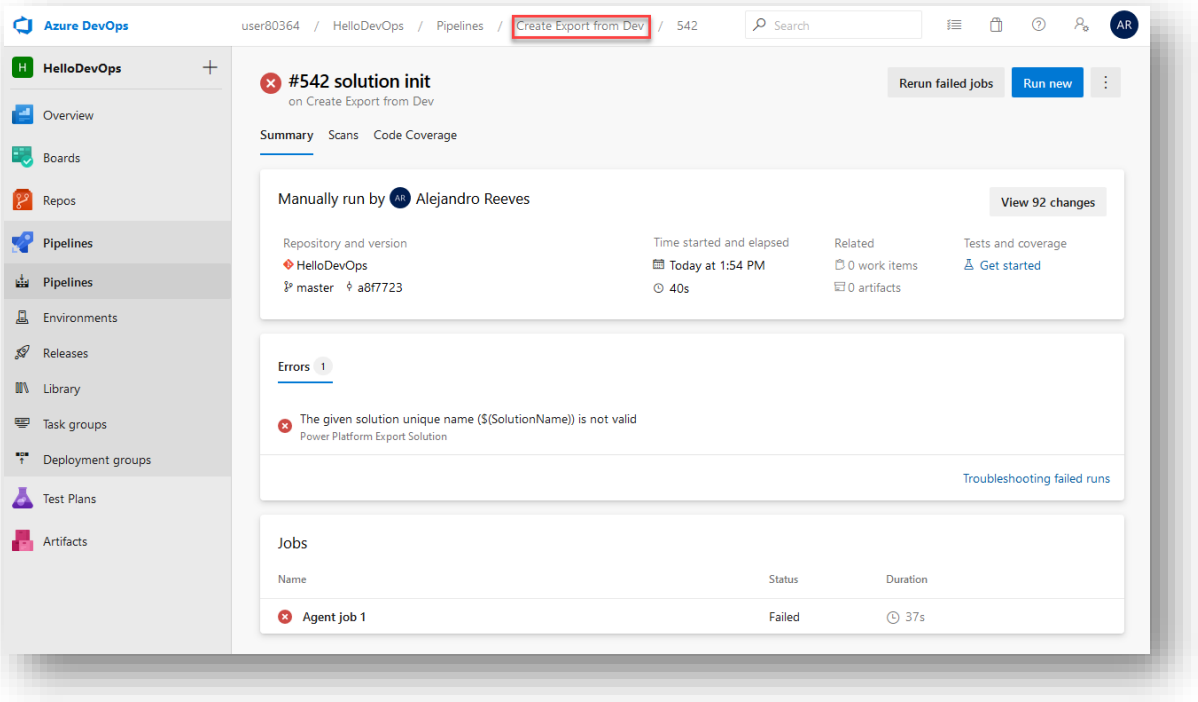
47. You can see your build has been queued. You can navigate to it directly in the notification on this screen or using the Builds area in the left navigation.



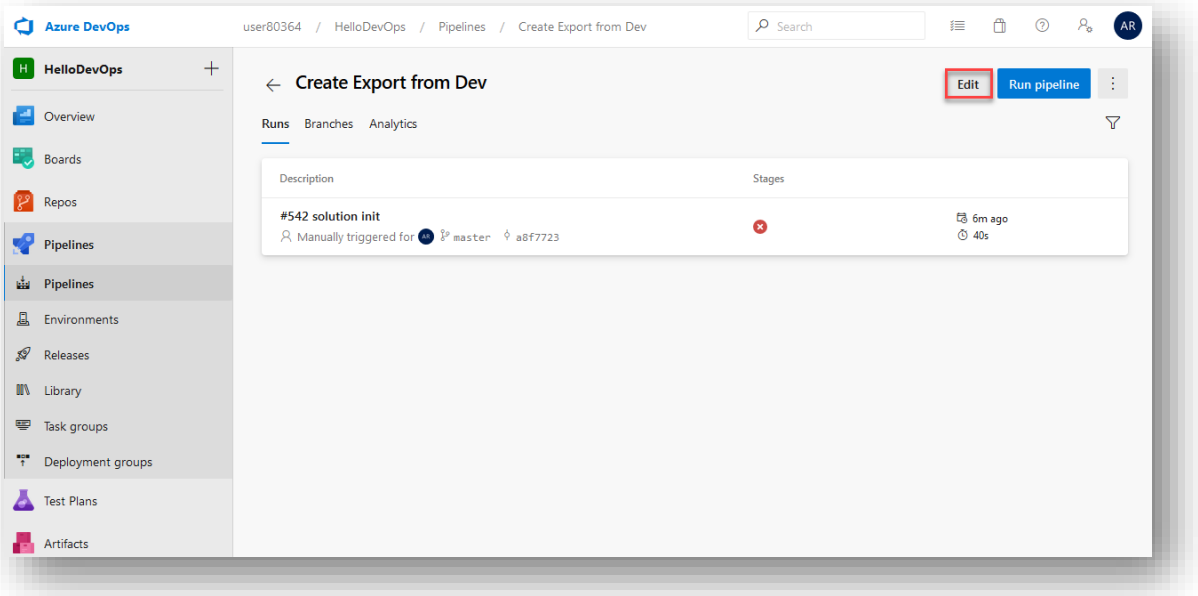
48. The desired outcome would be a series of green checkboxes, but if you have some issues with your pipeline, you should see the errors in the log. You can click on those for more details if you like. In our case, we have not defined our solution name variable yet, so the export step will fail. This was done intentionally to show you what a failure looks like and how to look at the details.



49. Go back to your pipeline by selecting the pipeline name from the top of screen. You also get there from clicking **Pipelines** in the left navigation and then select the pipeline you want to edit.

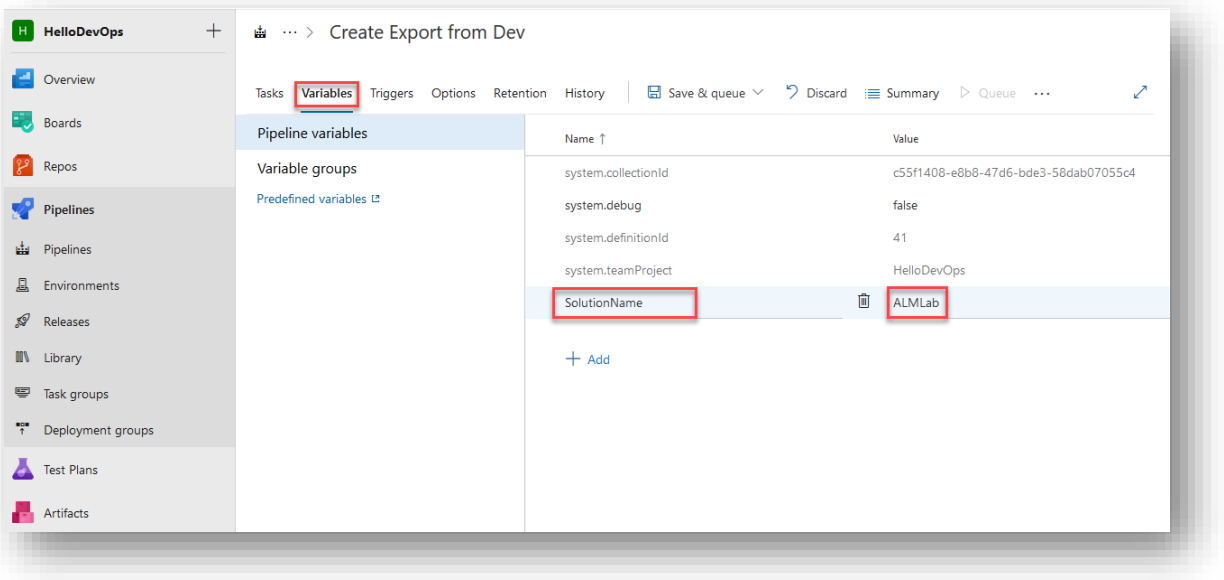


50. Click the **Edit** button.

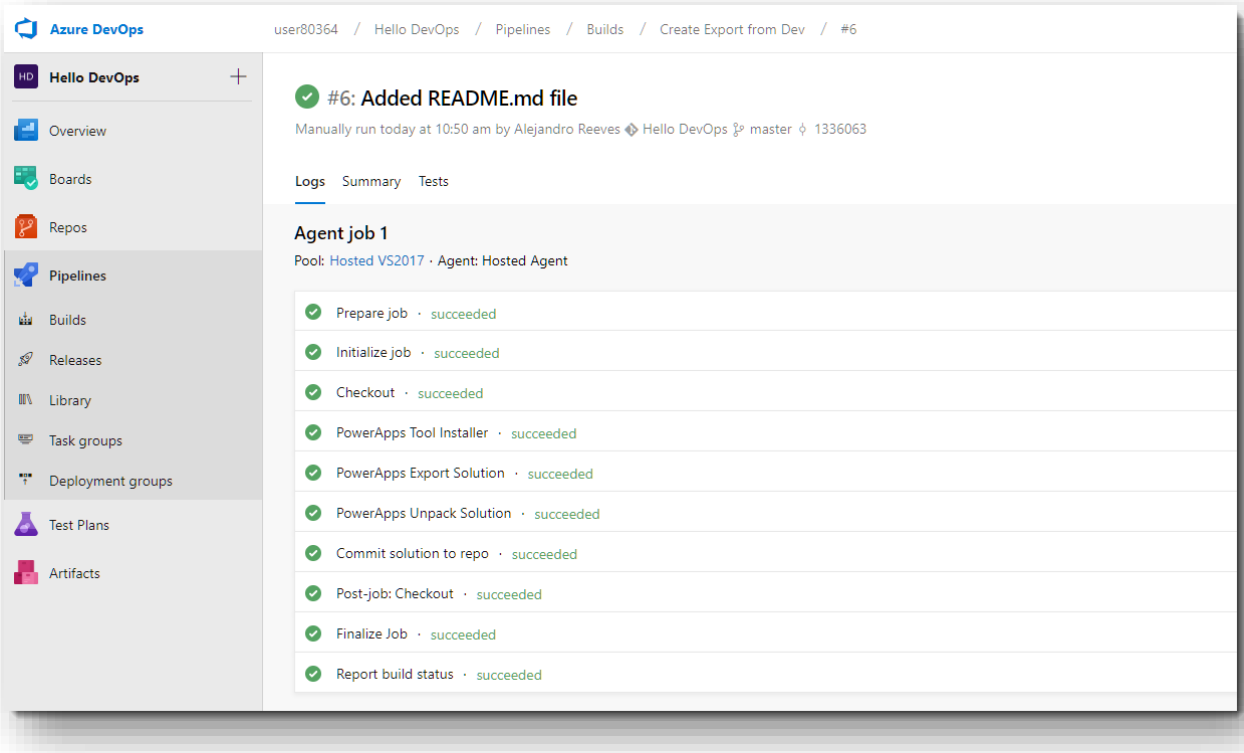


51. On the pipeline screen, switch to the **Variables** tab and click **Add** to add a new variable.

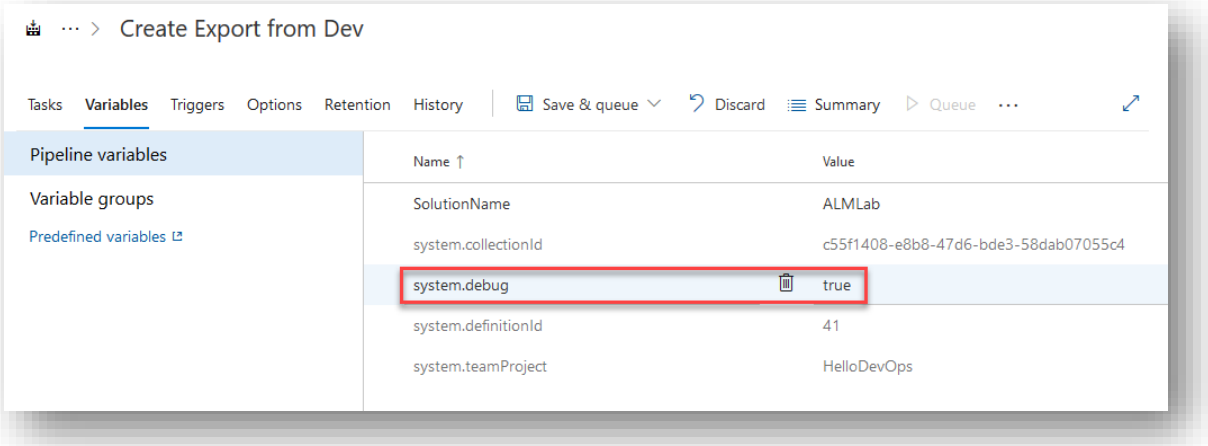
52. In the **SolutionName** field type your unique solution name as a value. It should be ALMLab if you are using the solution from Module 1.



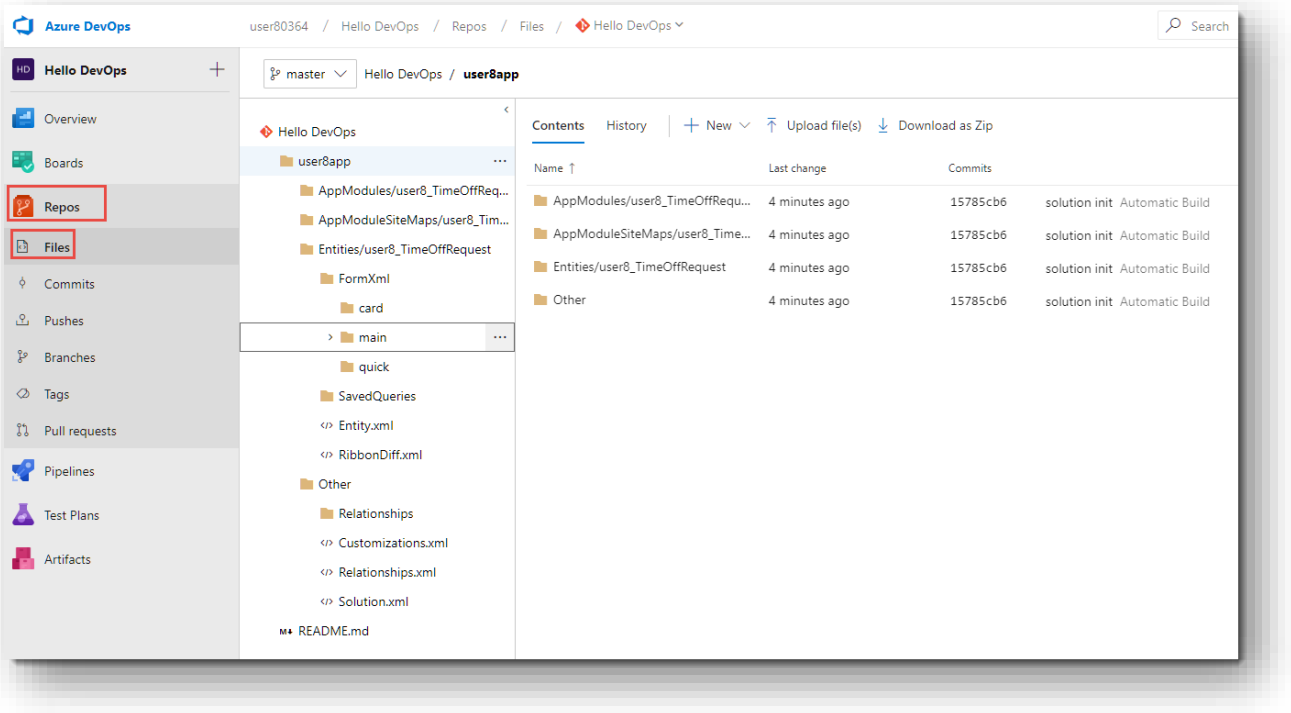
53. Click **Save & queue** and observe the results. Fix any issues until you get a successful build.



Note: You can get more details about any issues found by setting the variable system.debug to "true".



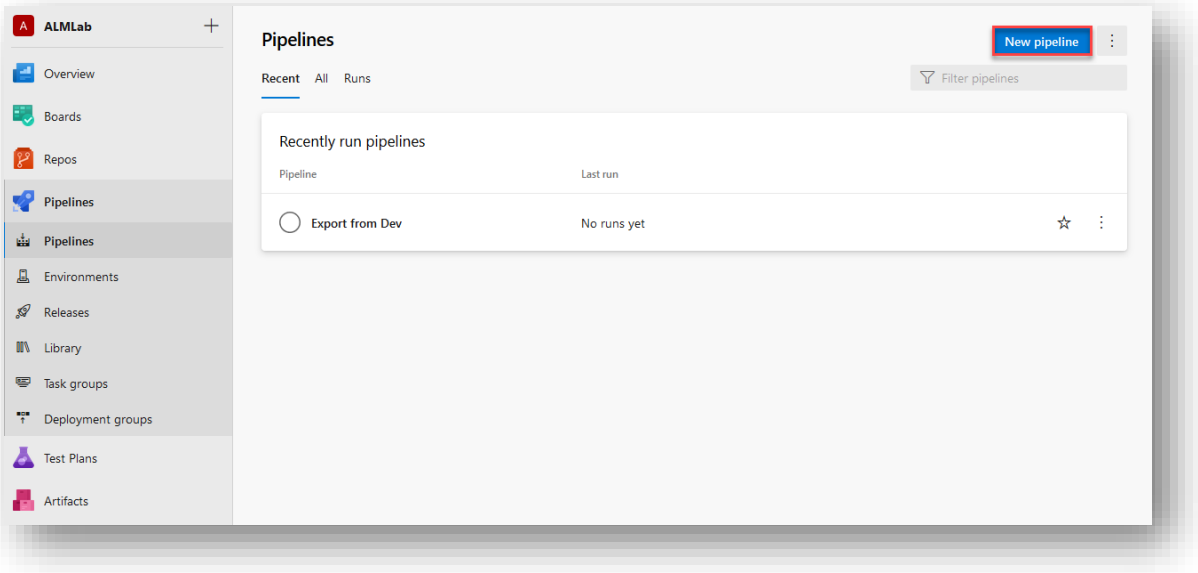
54. Let's go look at what was added to your source control now. Click **Repos** and **Files** and notice that it added a new folder that contains your solution files.



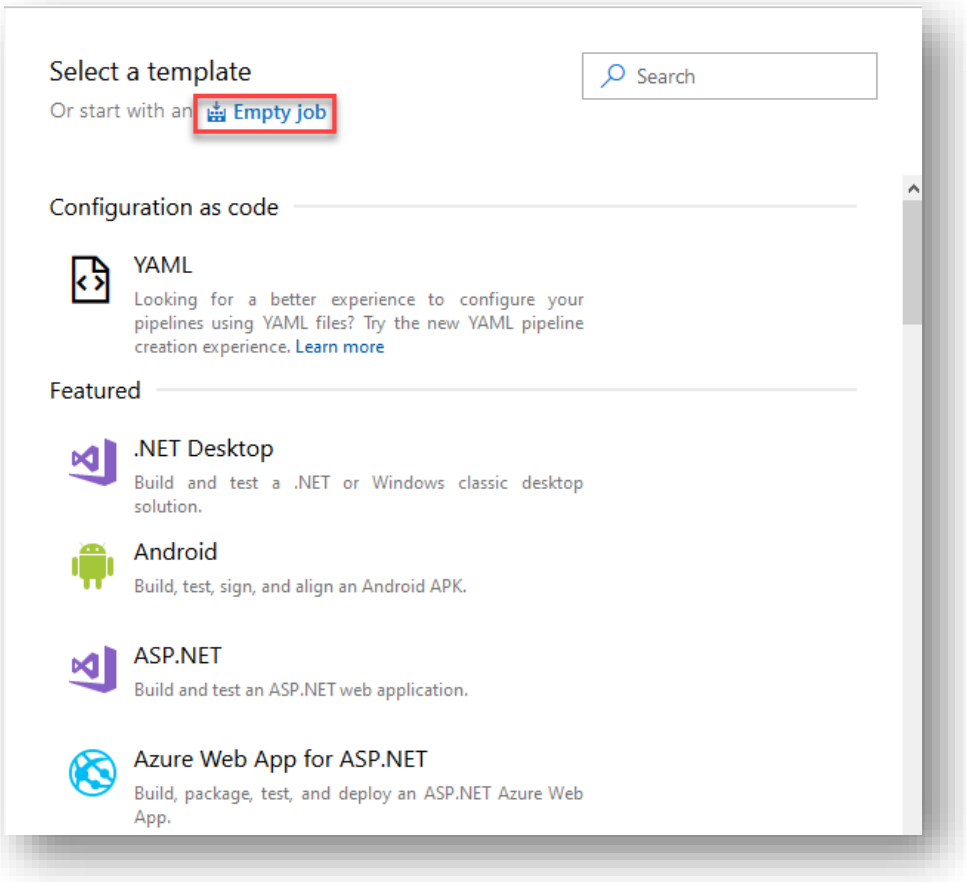
Pipeline 2: Build your Managed Solution

Your unmanaged solution is checked into source control, but you need to have a managed solution to deploy to other environments such as test and production. To obtain your managed solution, you should use a Just-In-Time (JIT) build environment where you would import your unmanaged solution files then export them as managed. These managed solution files will not be checked into source control but will be stored as a build artifact in your pipeline. This will make them available to be deployed in your release pipeline.

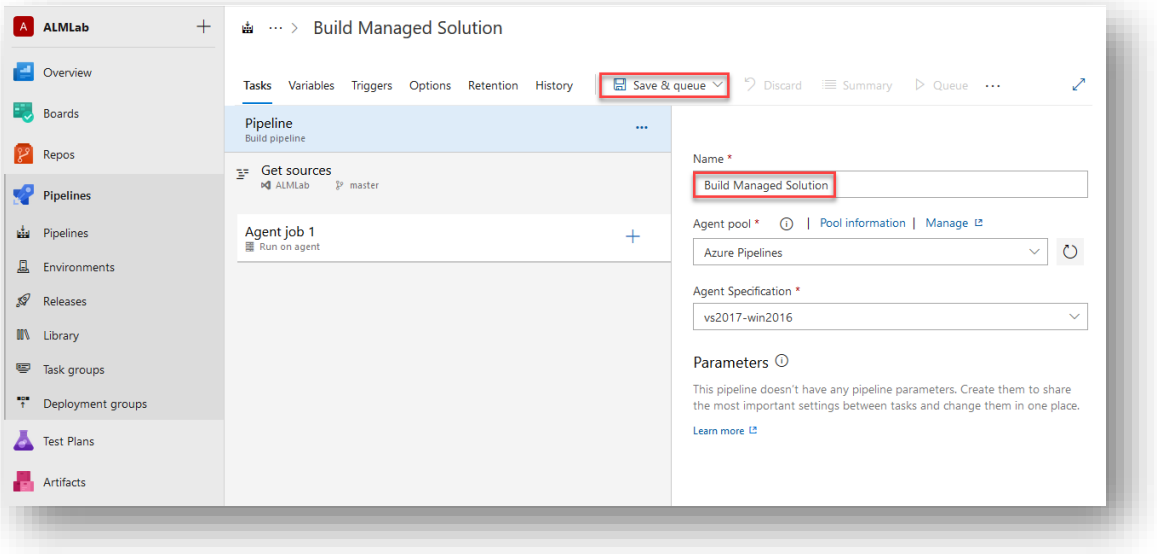
55. Navigate to your pipelines and add a new build pipeline.



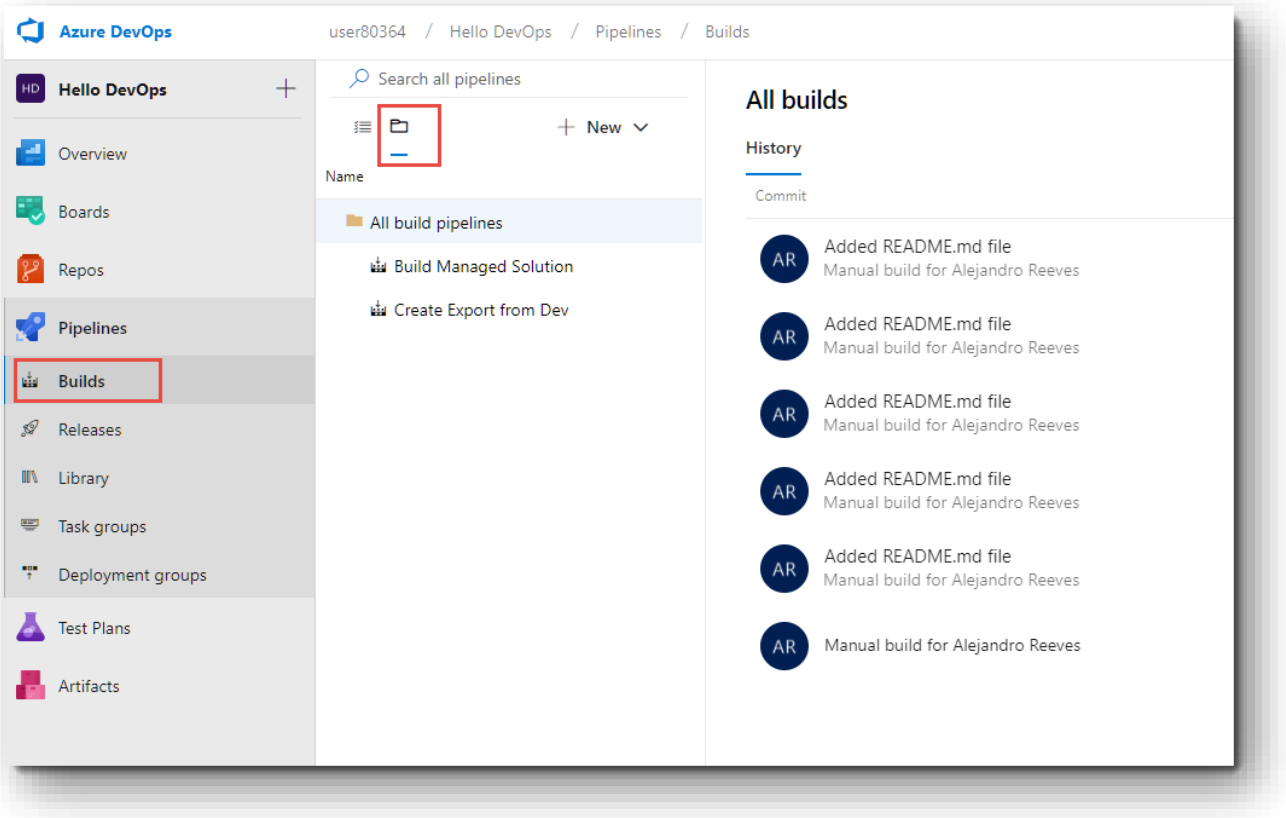
56. Click **Use the classic editor** link to build the pipeline without YAML. On the next screen use the defaults and click **Continue**. Finally, click on the **Empty jo**b link on the template selection page shown below.



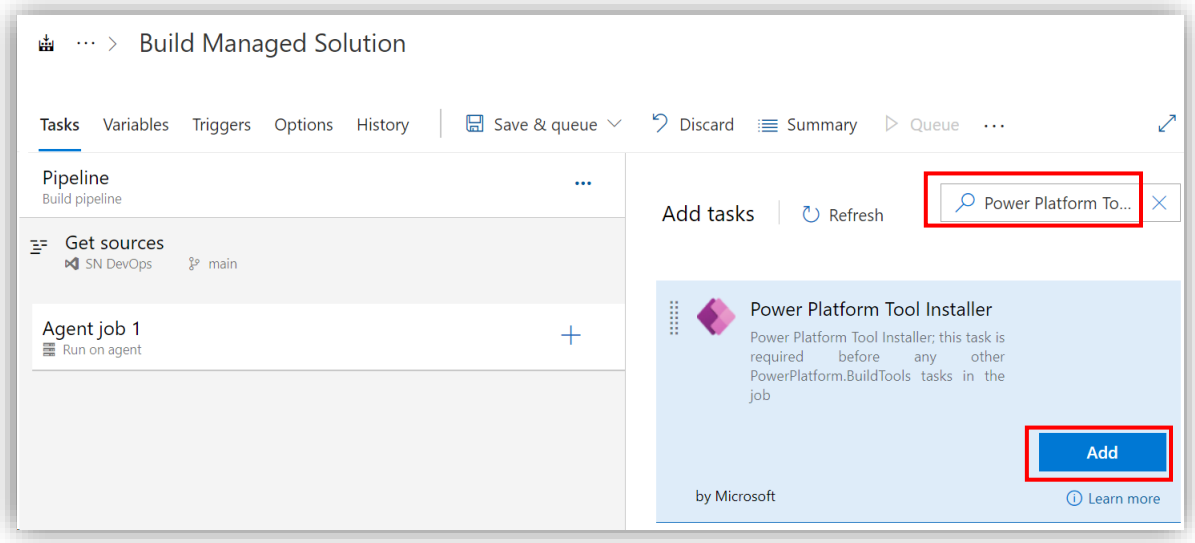
57. Name the pipeline “Build Managed Solution”’ and save it. This will give you an empty pipeline.



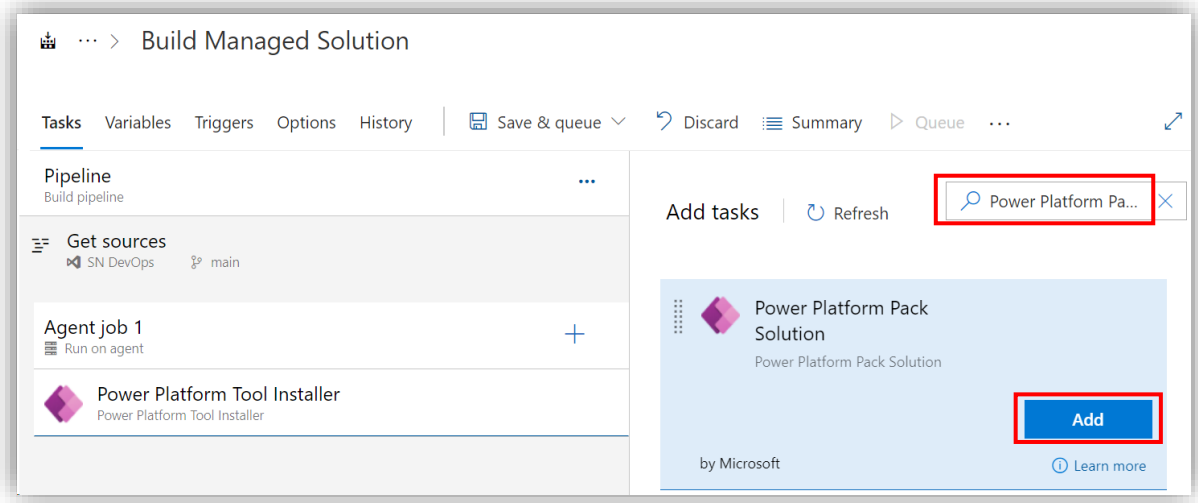
58. A small note on navigation. If you have problems seeing your newly created pipeline, you can click the little folder icon on the pipeline main page as the default view is only the recently used pipelines. If you have not used your pipeline yet and it does not show up in the list, change the view to all pipelines and you should be able to see and interact with it.



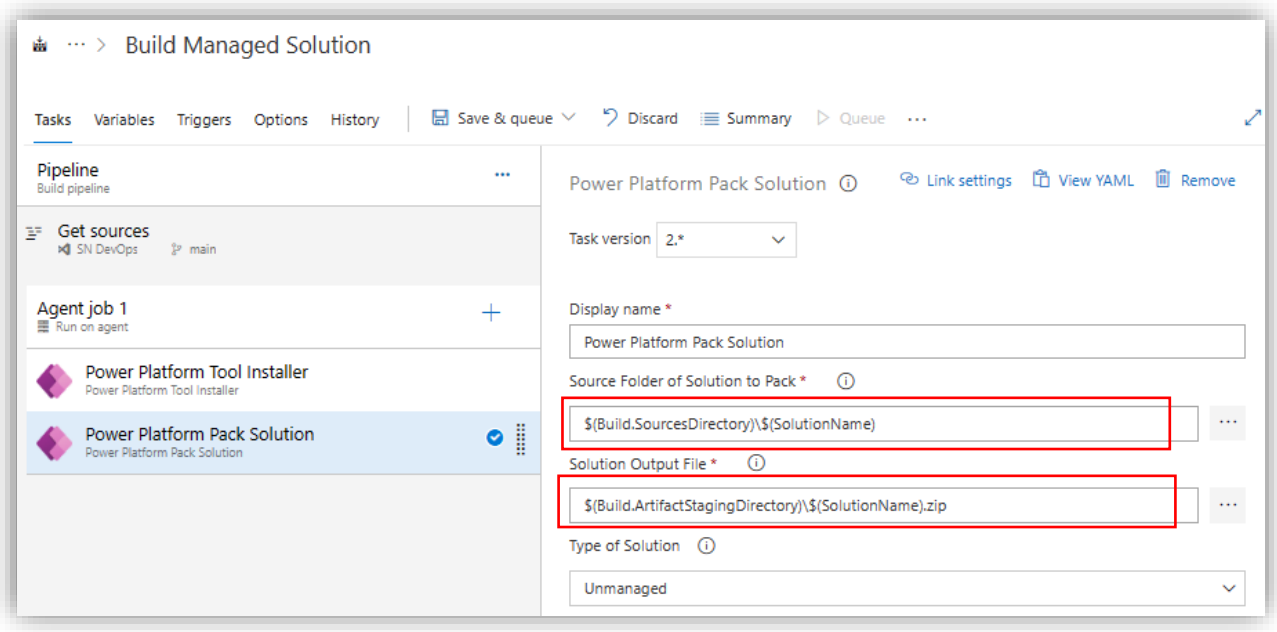
59. In your empty pipeline, add the **Power Platform Tool Installer** task.



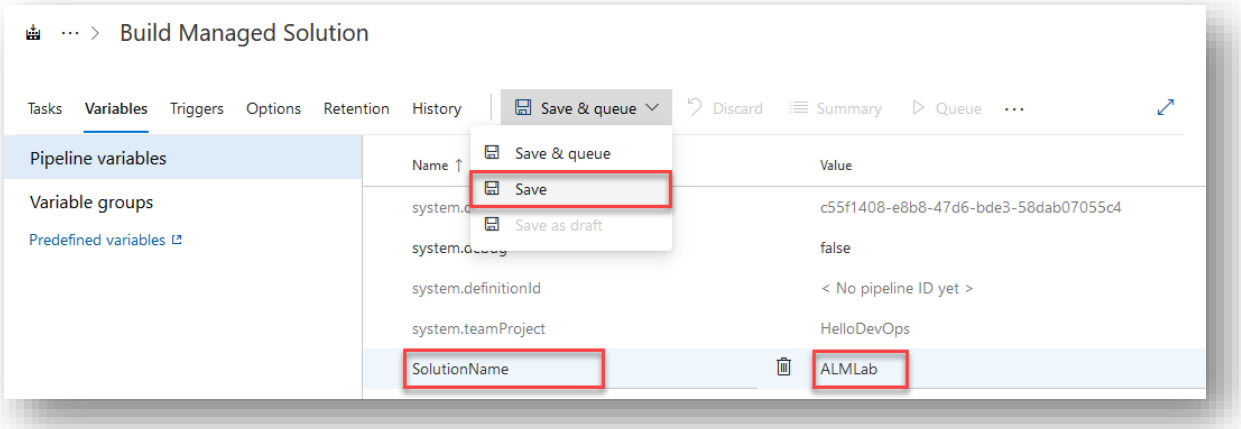
60. Add the **Power Platform Pack Solution** task.



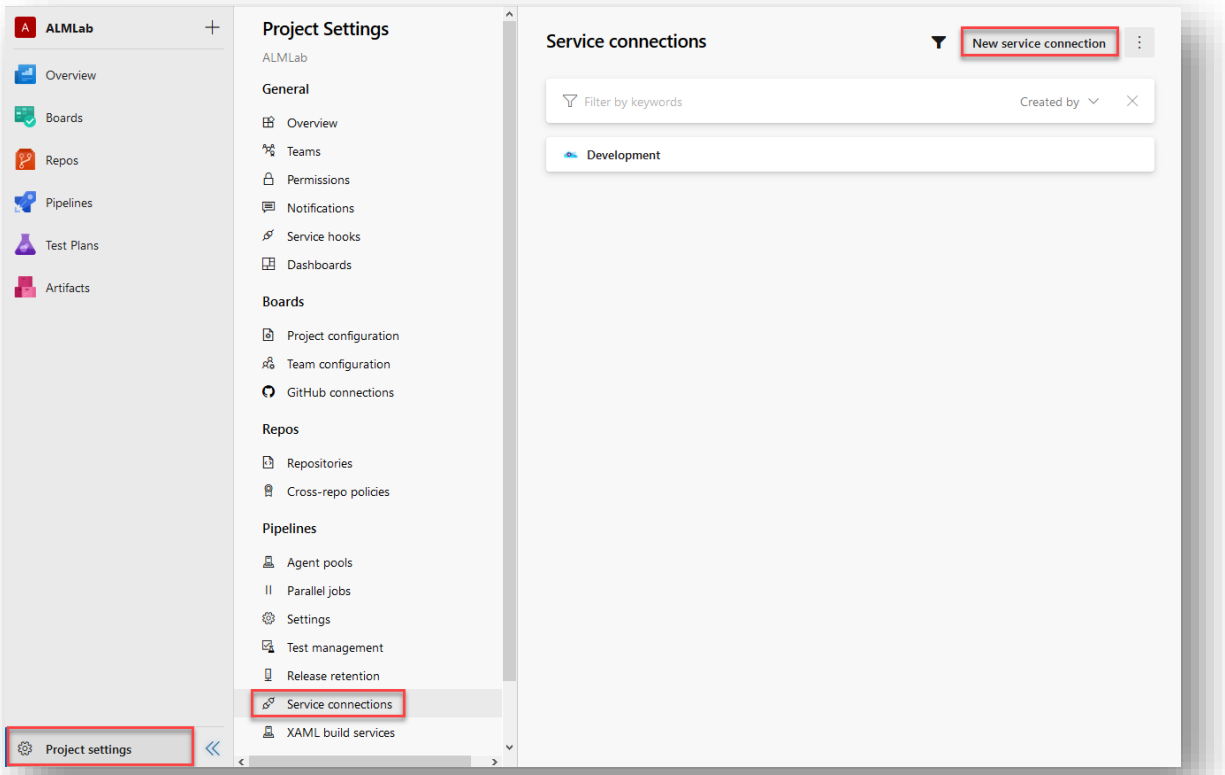
61. Configure the pack solution task with the following parameters:
    - Source Folder of Solution to Pack: **$(Build.SourcesDirectory)\$(SolutionName)**
    - Solution Output File: **$(Build.ArtifactStagingDirectory)\$(SolutionName).zip**



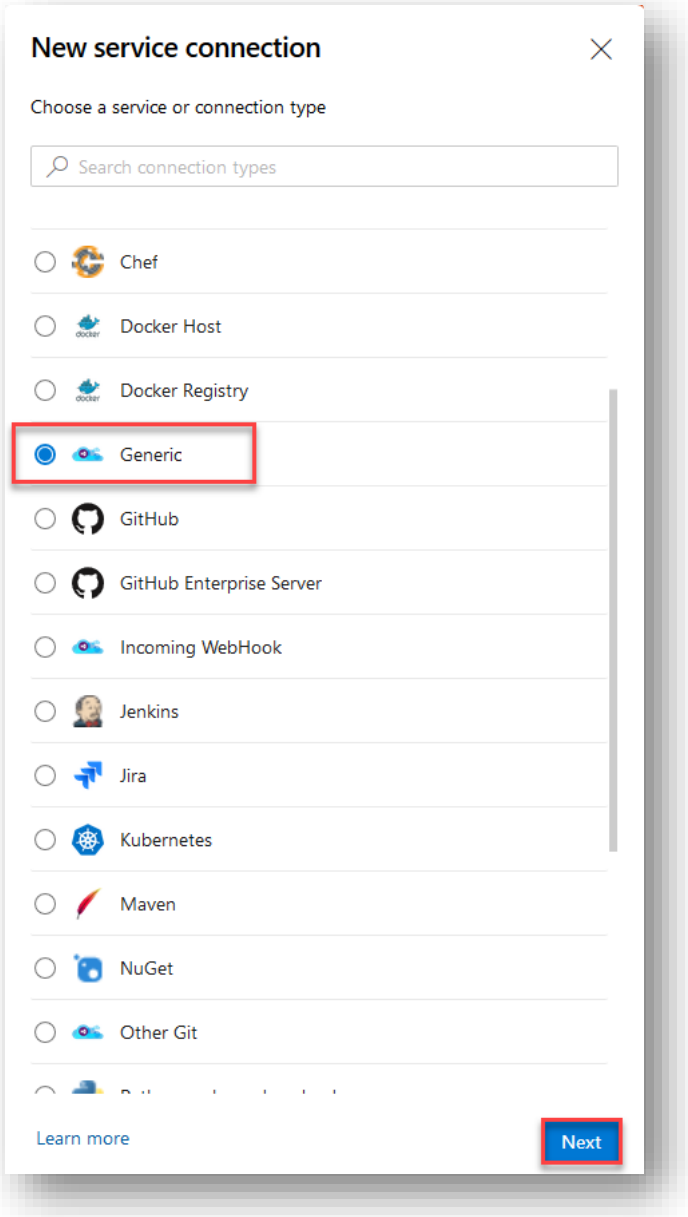
62. On the pipeline screen, switch to the variables tab in the pipeline and add the SolutionName variable with your unique solution name. This should be ALMLab if you are using the solution from Module 1. Click **Save** (through **the Save & queue** dropdown).



63. The next task will be to import the solution into your build server, so we need to add a connection to that environment before we add and configure the task to do the import. Click the **Project settings** link in the lower left of the screen and navigate to **Service connections**, then click **New service connection**.

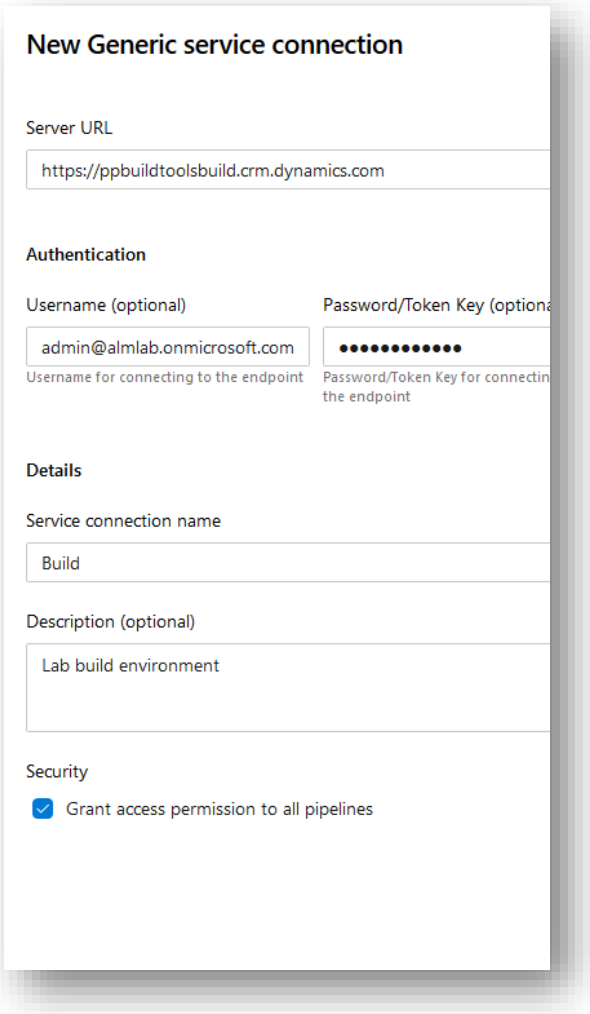

ALMLab
Overview
Boards
Repos
Pipelines
Test Plans
Artifacts
Project settings
Project Settings
ALMLab
General
Overview
Teams
Permissions
Notifications
Service hooks
Dashboards
Boards
Project configuration
Team configuration
GitHub connections
Repos
Repositories
Cross-repo policies
Pipelines
Agent pools
Parallel jobs
Settings
Test management
Release retention
Service connections
XAML build services
Service connections
New service connection
Filter by keywords
Created by
Development

64. Select **Generic** on the **New service connection** page and click **Next**.

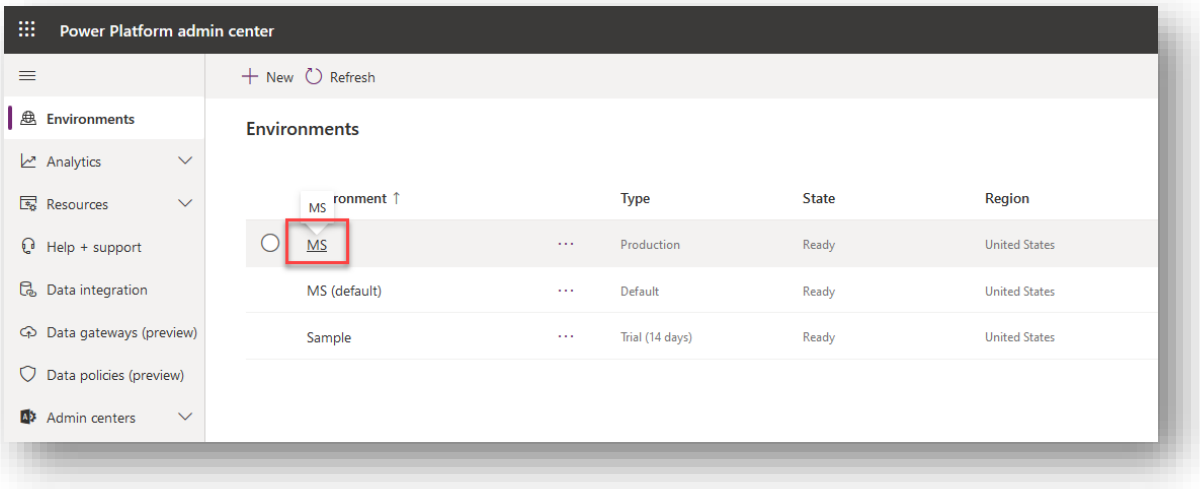


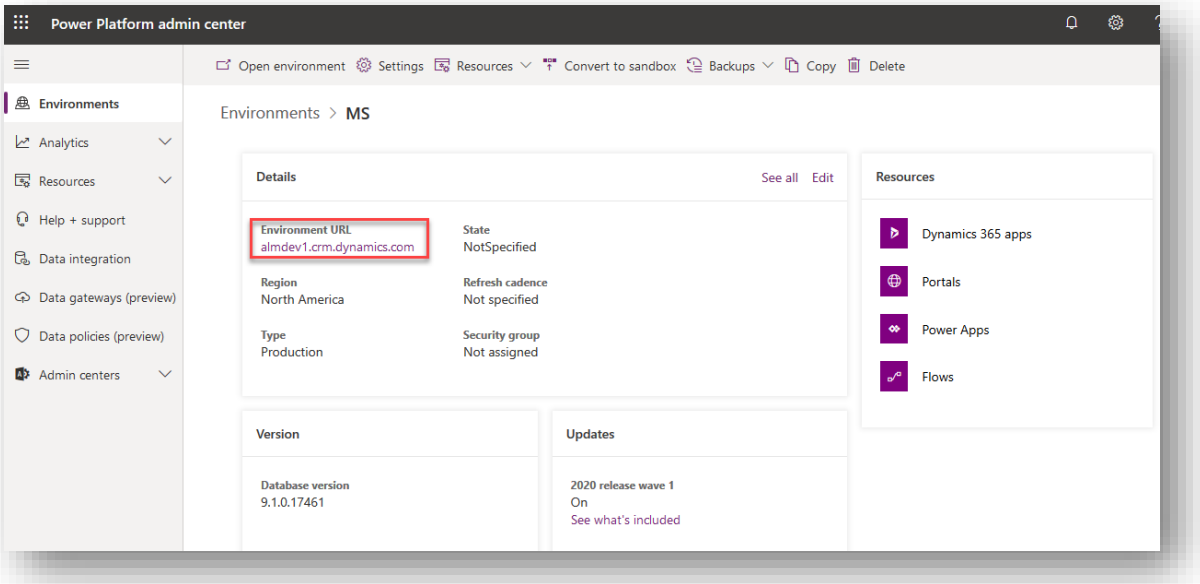
65. Fill out the required details:
    - **Server URL**: https://<environment-url>.crm.dynamics.com. This should be the URL to your build environment
    - **Username:** Username of a user with administrator access to the environment
    - **Password:** Associated password for the user
    - **Service connection name:** This name will be used to identify which environment to connect to in the Build tasks
    - **Description:** Give the connection a description that helps you identify the environment the service connection connects to

    The sample connection is shown below.



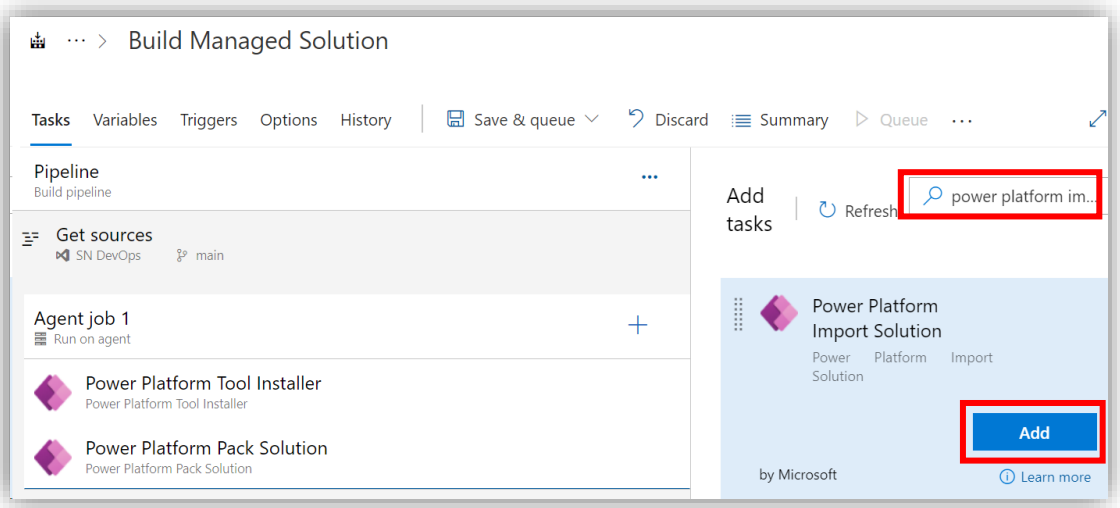
Note: If you forgot to record it the server URL for your build environment, simply visit https://admin.powerplatform.microsoft.com and click the environment. This will open another screen displaying the environment URL. Make sure it is your build environment.



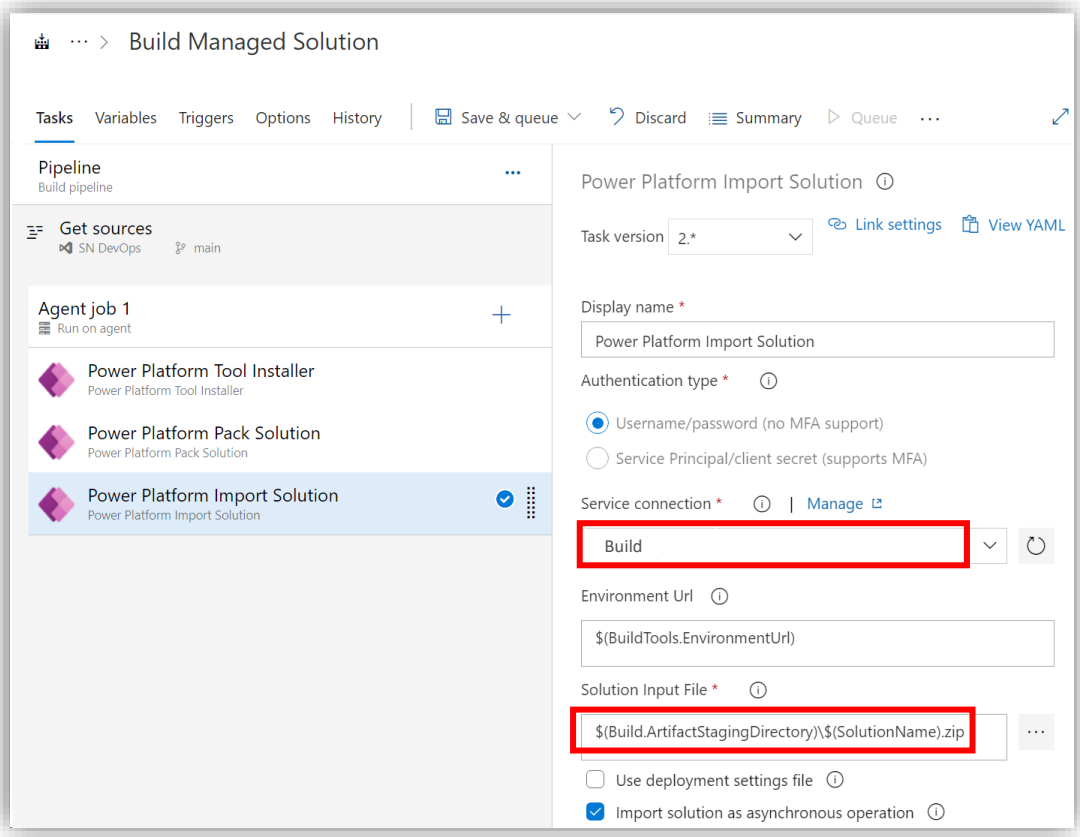


66. Click **Save**. You will be in the pipeline service connections area in your project.

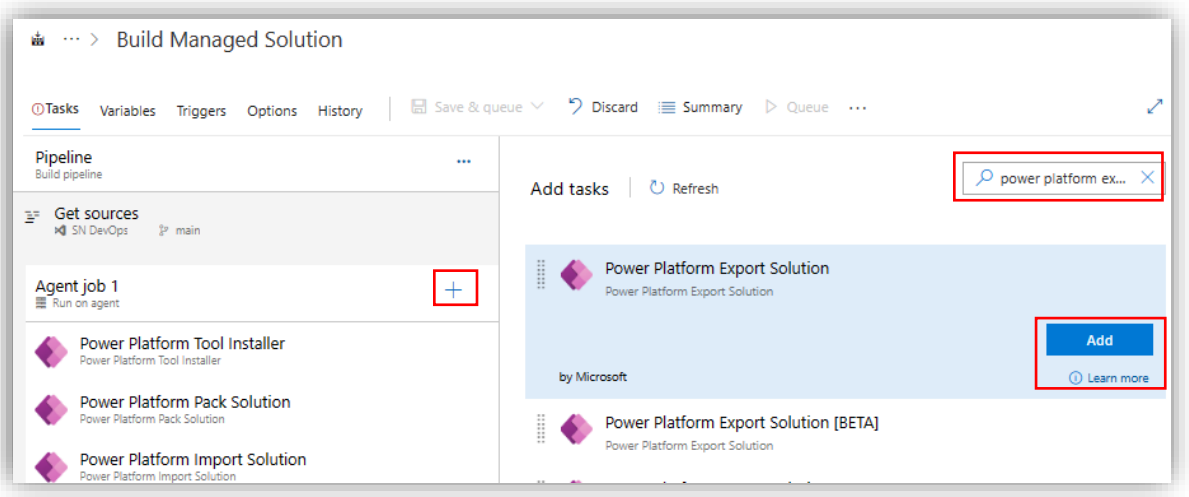
67. Navigate back to your Build Managed Solution pipeline (If you don't see the pipeline go to step 57 on how to view all pipelines). Click **Edit**, and then add the **Power Platform Import Solution** task to take the solution file and import it into your build environment.



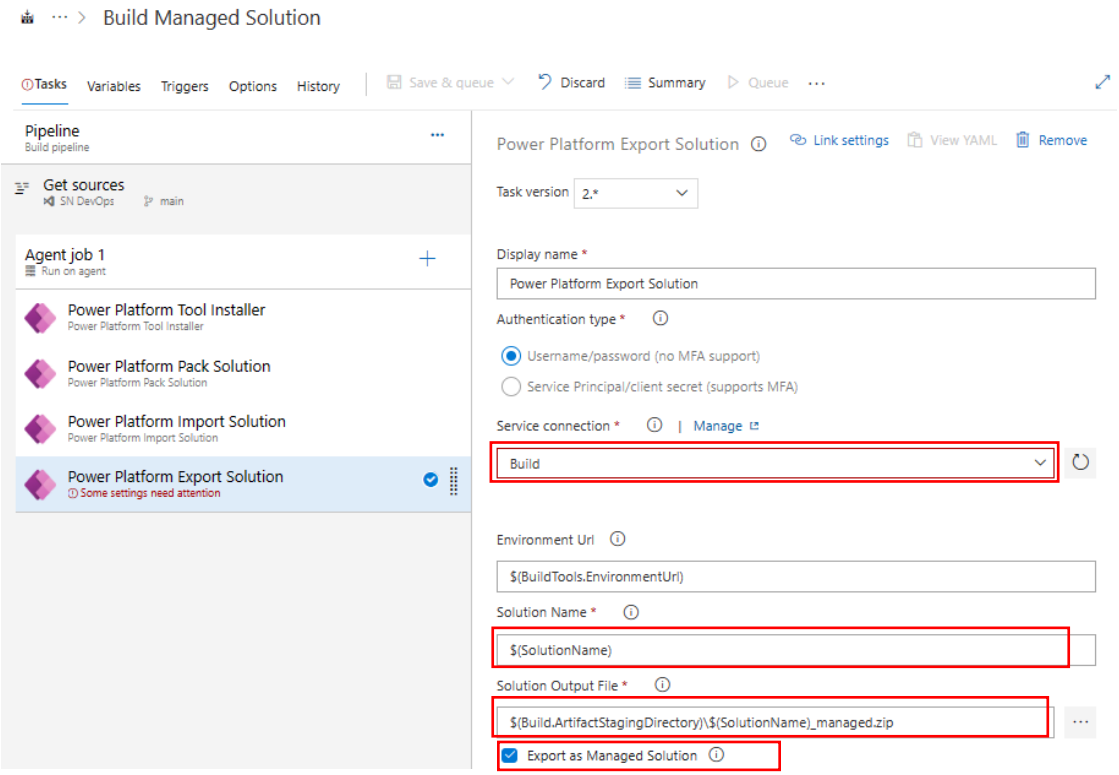
68. Select the connection you just added to your build server and provide the following for the solution input file: **$(Build.ArtifactStagingDirectory)\$(SolutionName).zip.**



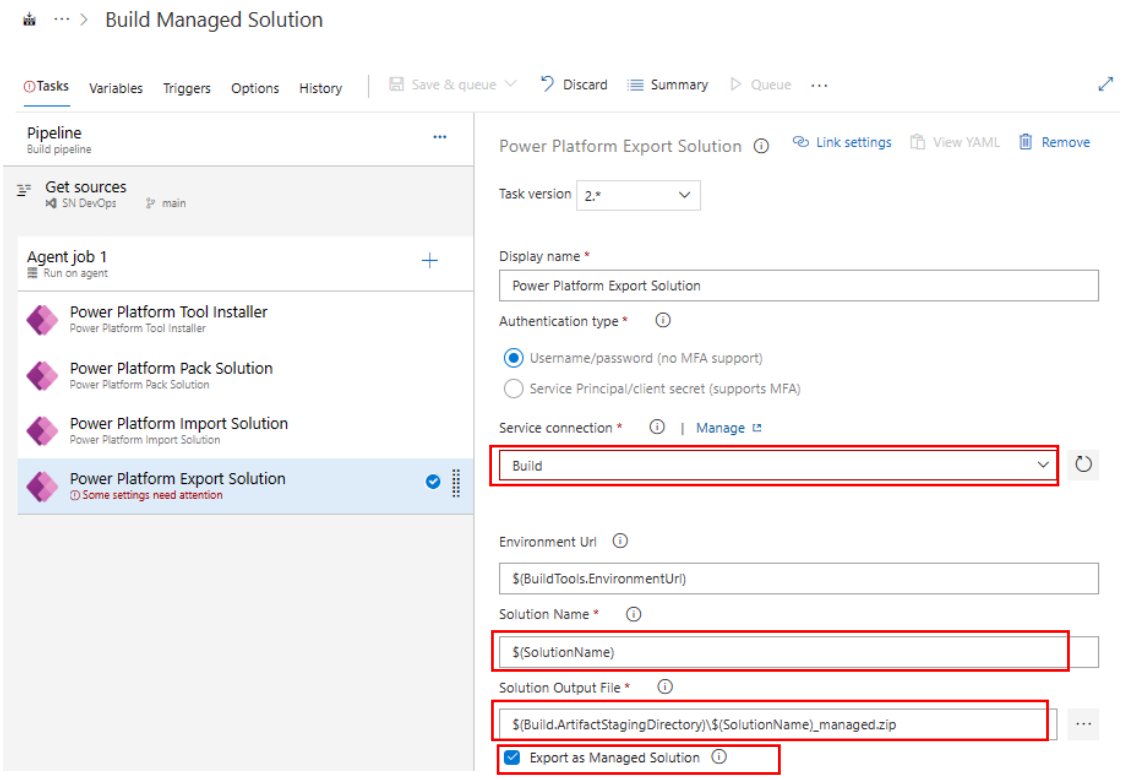
69. Add a task to export the solution file as a managed file.



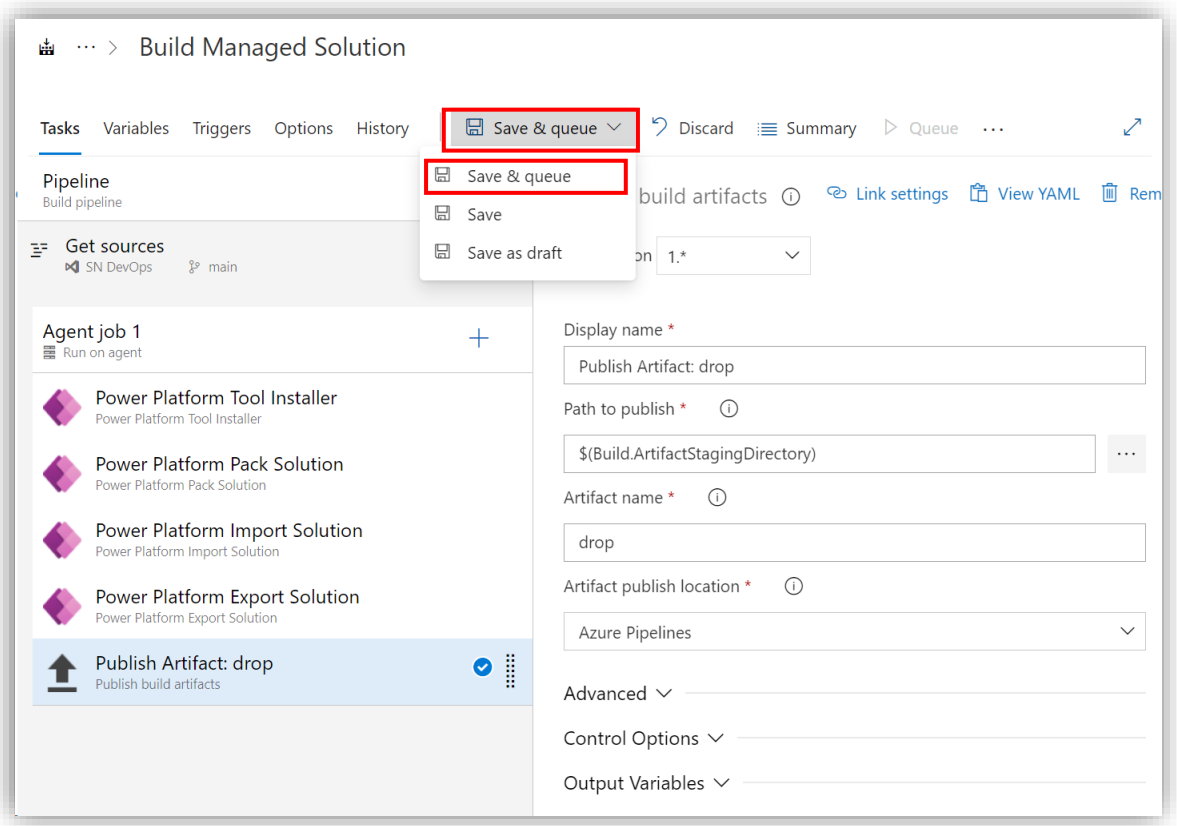
70. Select your build environment, check the **Export as managed solution** checkbox and enter the following details:
Solution Name: $(SolutionName)
Solution Output File: $(Build.ArtifactStagingDirectory)\$(SolutionName)_managed.zip



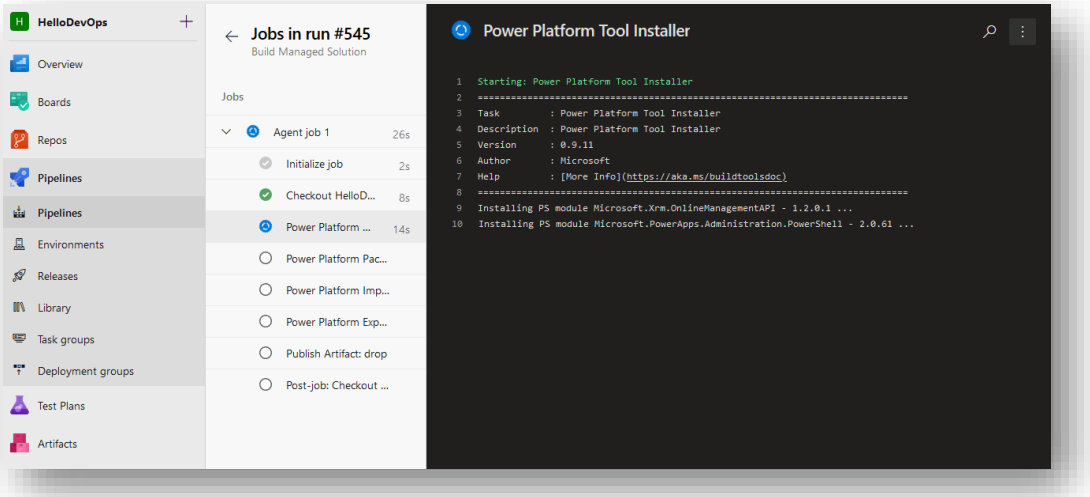
71. The files at this stage will be in the build agent and we need to publish it as a build artifact. Add a new pipeline step **Publish Pipeline Artifact**.



72. Leave the predefined values as-is and click **Save & queue**:

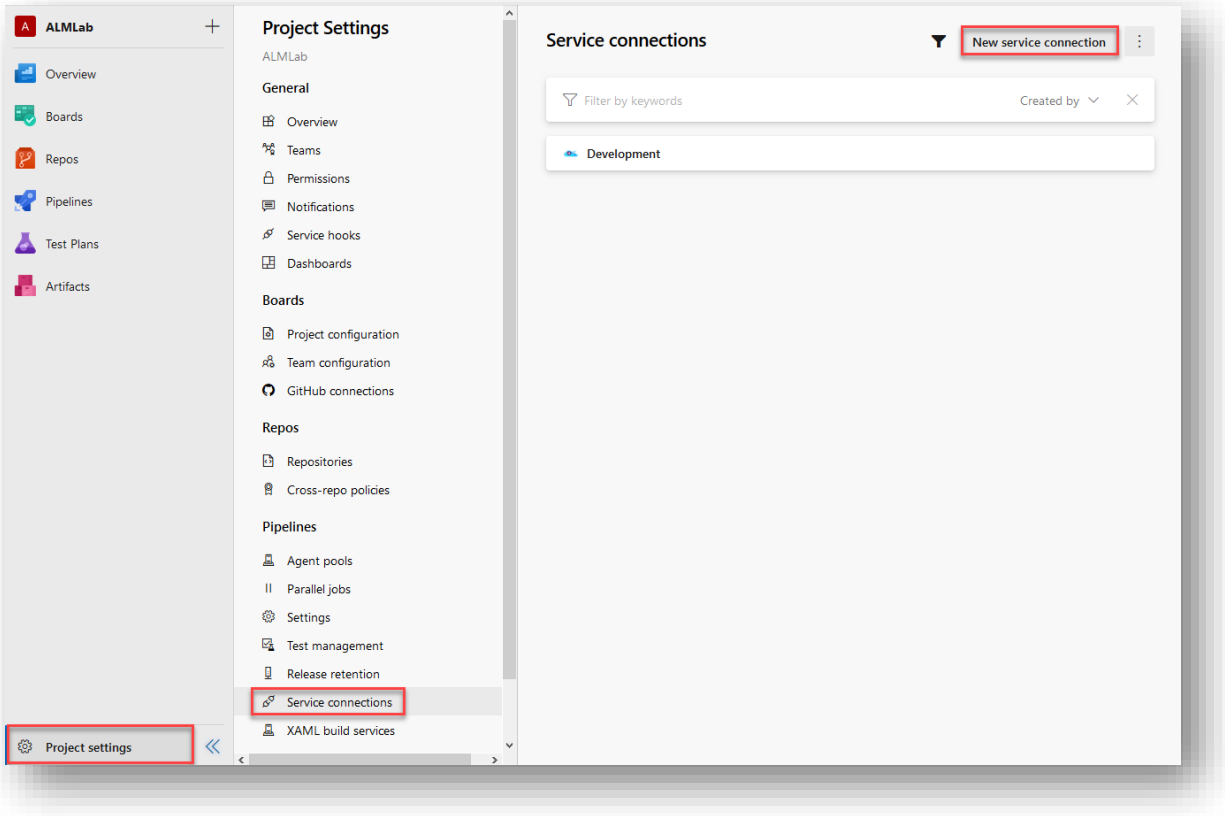


73. Monitor it until it is successful.

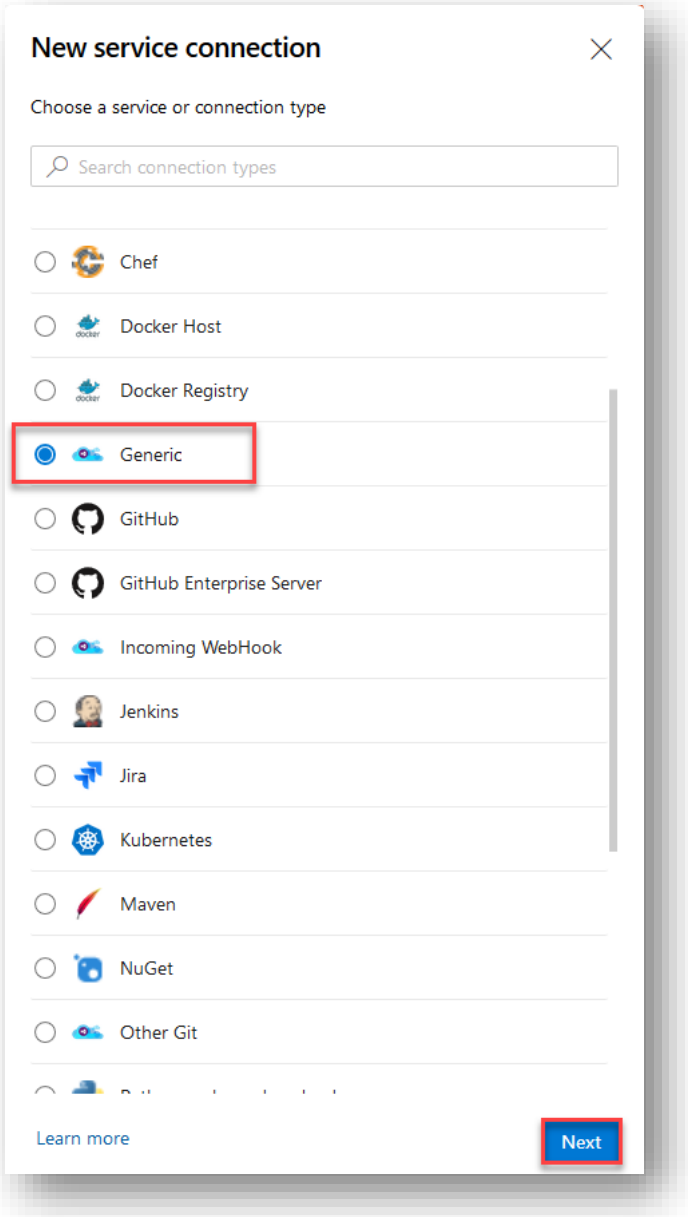


Release Pipeline: Release to Production

74. The next task will be to import the solution into your production server, so we need to add a connection to that environment before we add and configure the task to do the import. Click the **Project settings** link in the lower left of the screen and navigate to **Service connections**, then click **New service connection**.

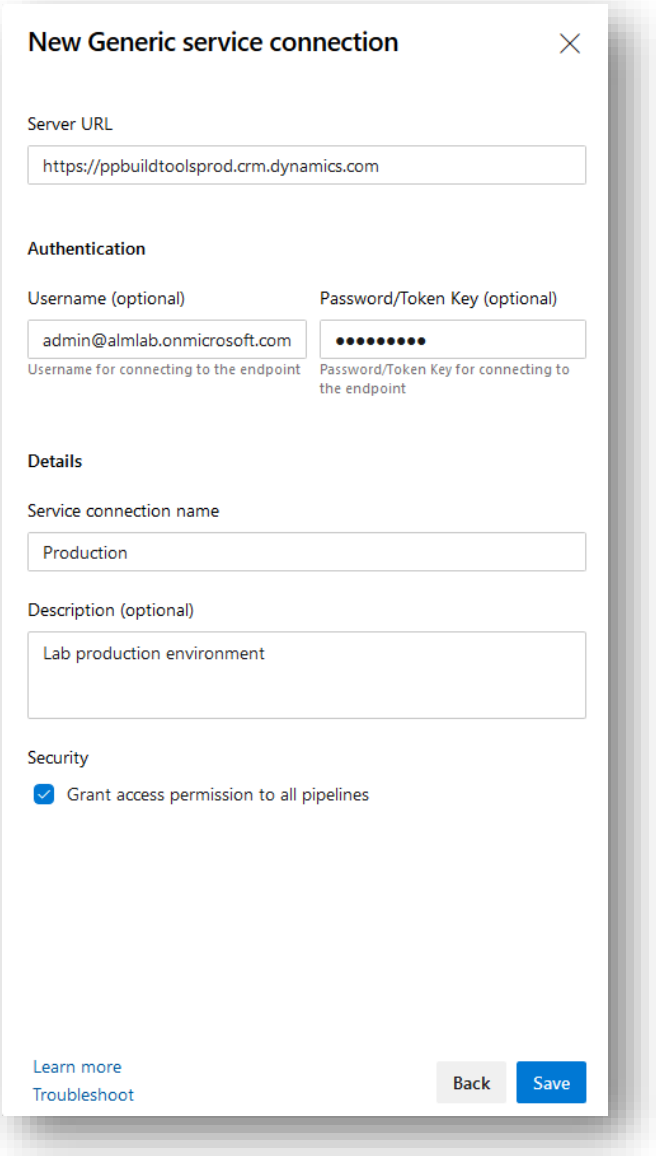


75. Select **Generic** on the **New service connection** page and click **Next**.



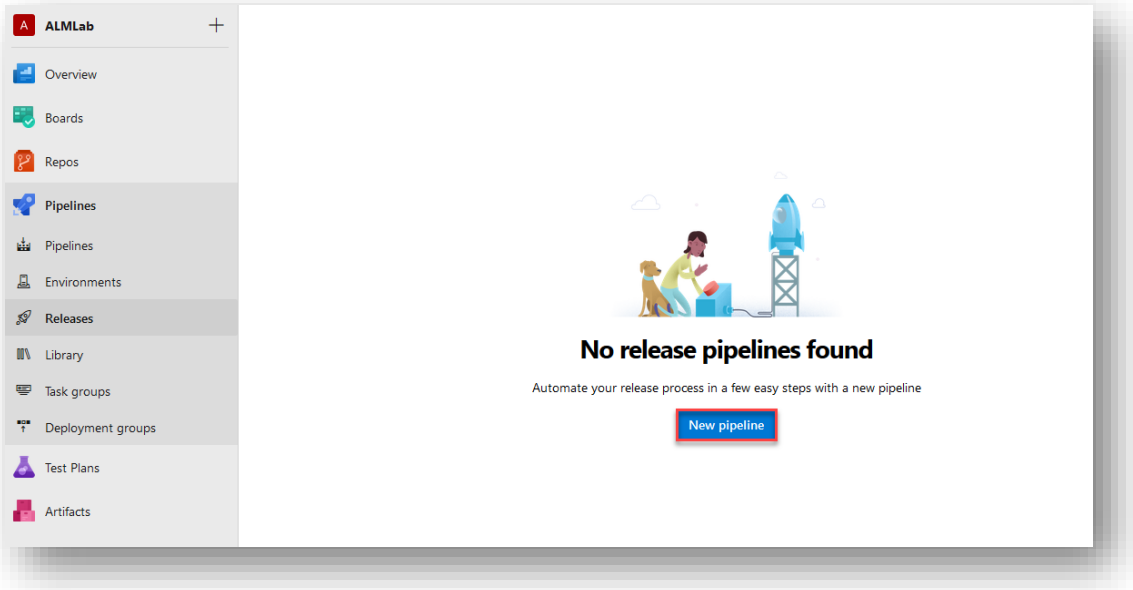
76. Fill out the required details:
    - **Server URL**: https://<environment-url>.crm.dynamics.com. This should be the URL for your production environment.
    - **Username:** Username of a user with administrator access to the environment
    - **Password:** Associated password for the user
    - **Service connection name:** This name will be used to identify which environment to connect to in the build tasks
    - **Description:** Give the connection a description that helps you identify the environment the service connection connects to

    The sample connection is shown below.

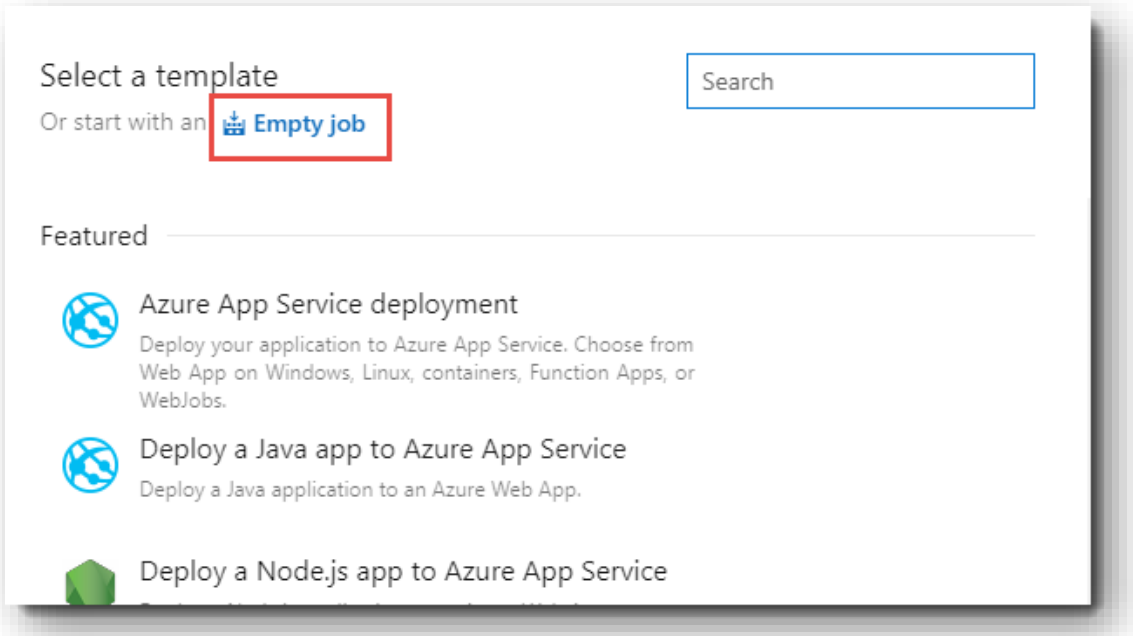


77. Click **Save**. You will be in the pipeline service connections area in your project.

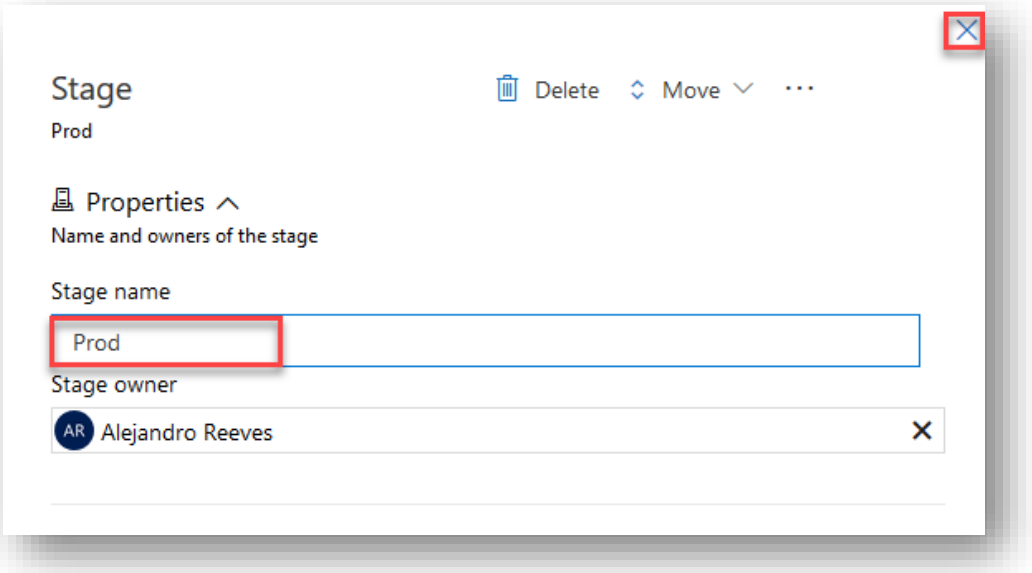
78. To deploy a build, you will configure a release pipeline. Navigate to the release pipelines. You likely won't have one yet, so click on the **New pipeline** button.



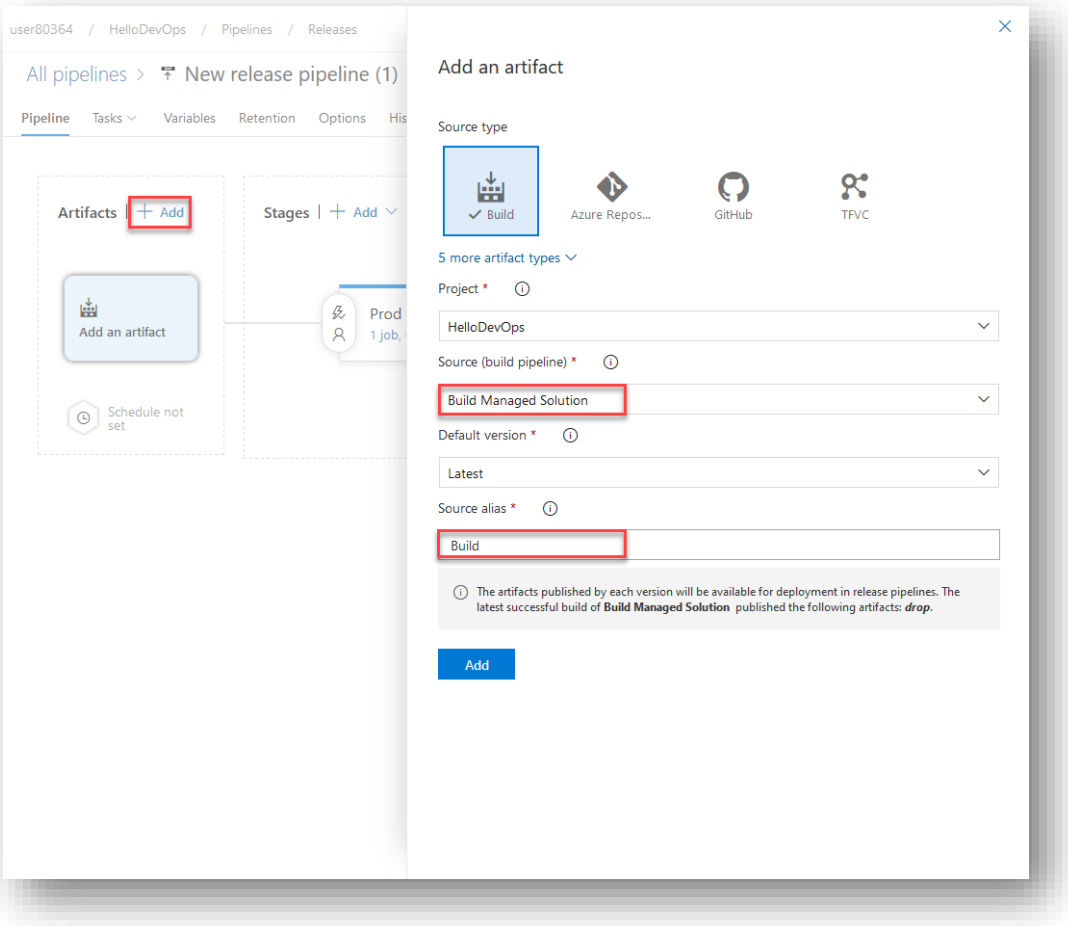
79. Click on **Empty job** when selecting a template.



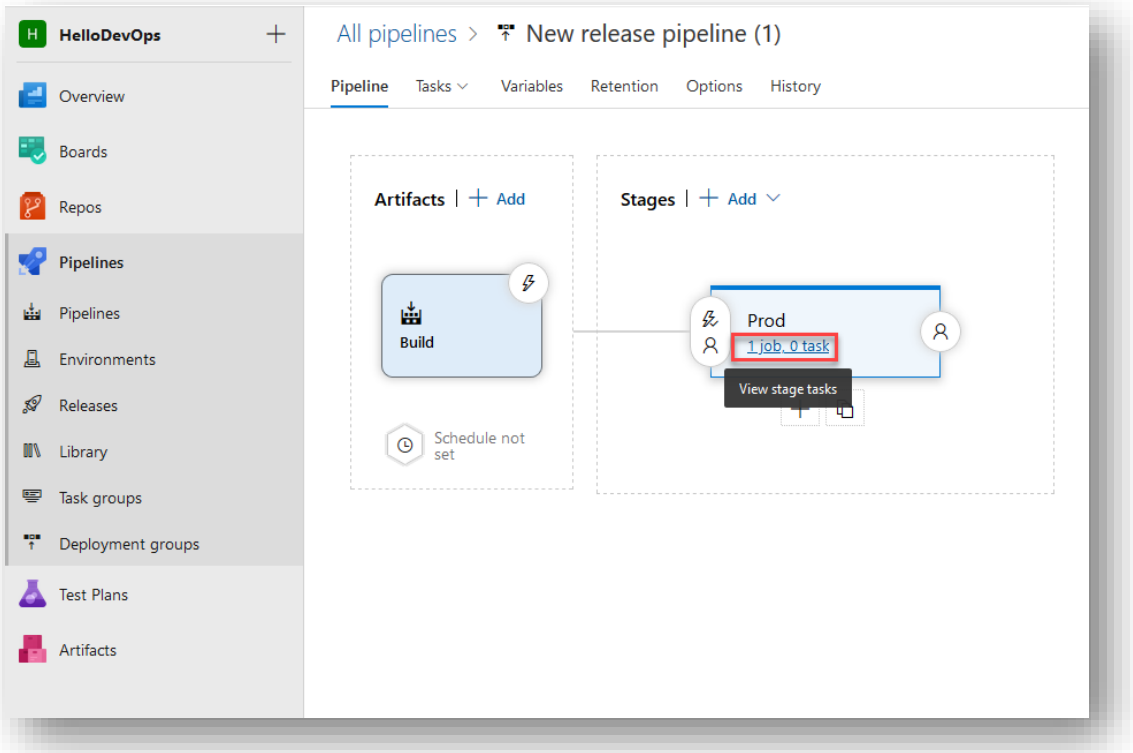
80. Give it a stage name and then close the **Stage** dialogue.



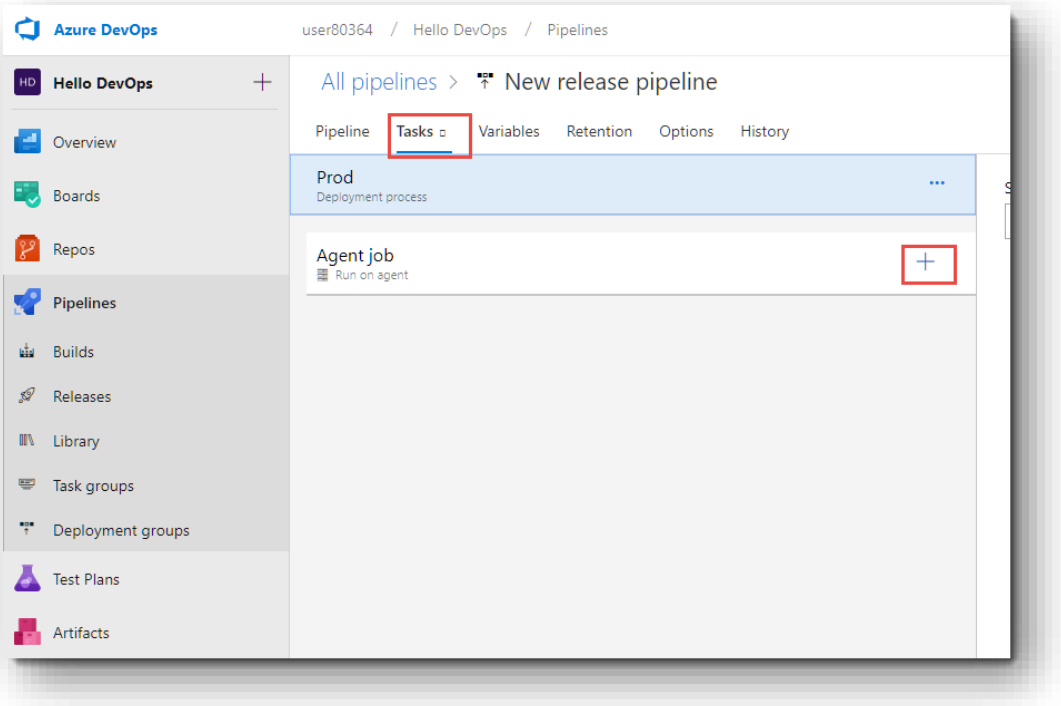
81. Next, we need to add the artifact that will be used in the deployment. Click the **Add** button in the **Artifacts** list. In the **Add an artifact** screen, select your "Build Managed Solution" from the source build pipeline. To make it easier to not have to deal with encoding spaces in filenames, change your source alias to something simple, like "Build". Leave the rest as default. Click **Add** when completed.



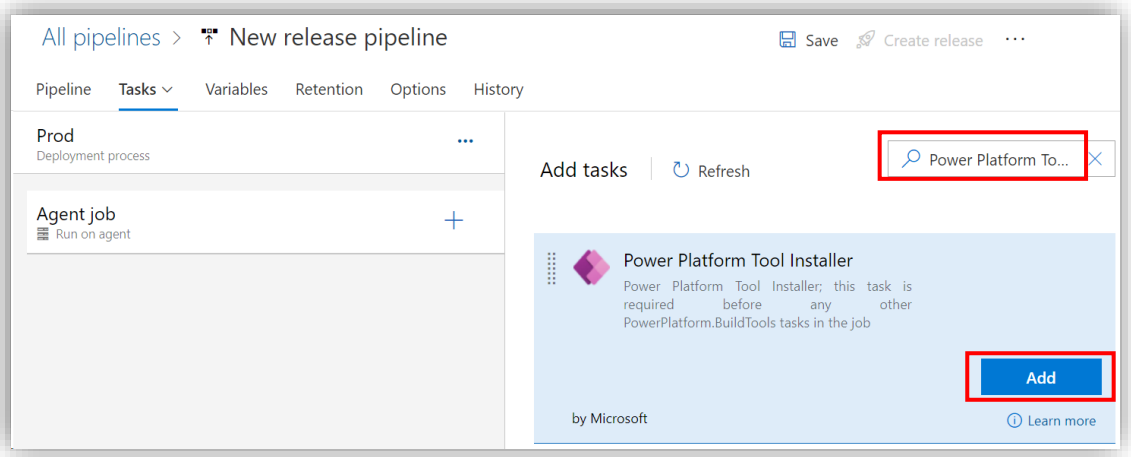
82. Switch to the **Tasks** view and click the plus **+** button to add a new task.



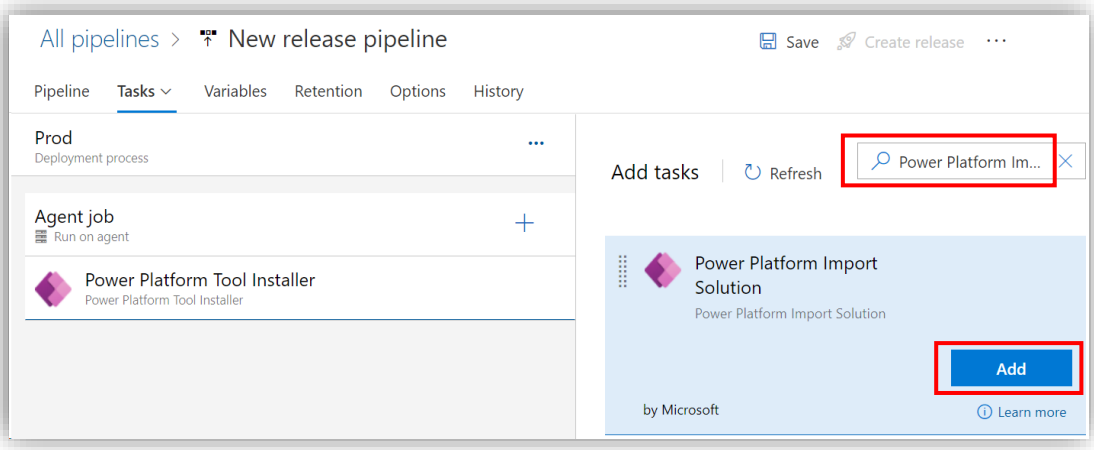
83. In the **Tasks** view, click the plus **+** button to add a new task.



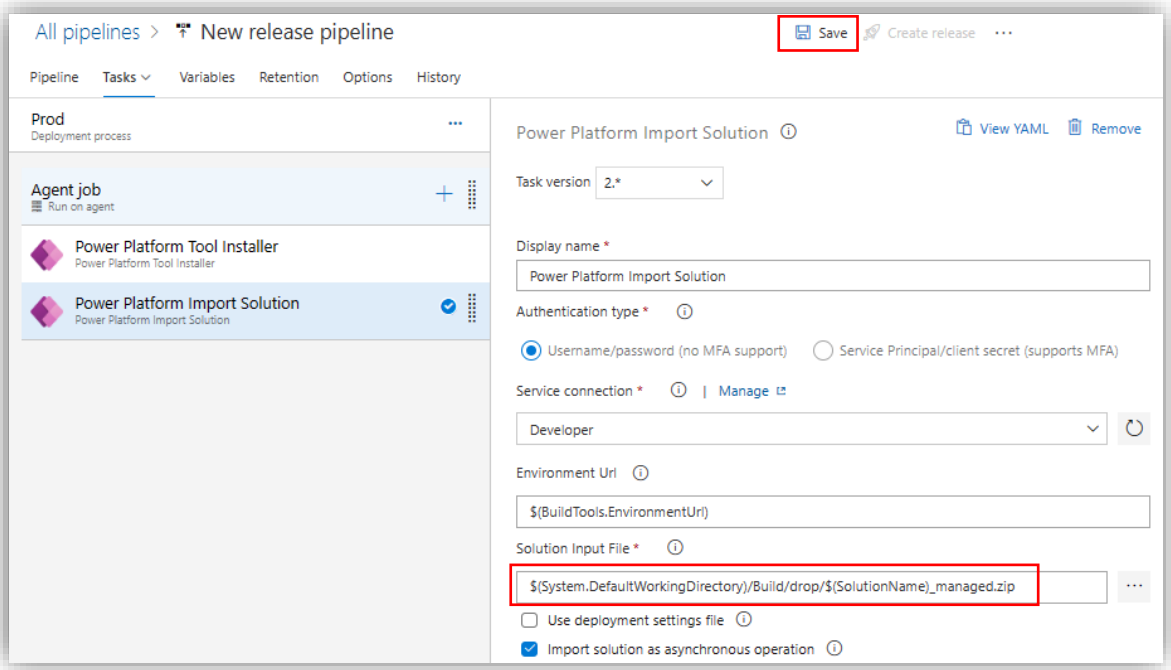
84. Add the **Power Platform Tool Installer** task.



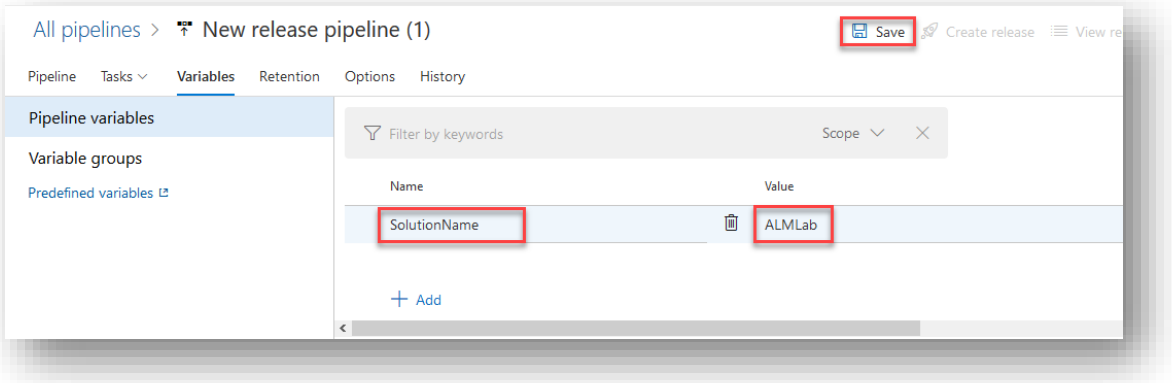
85. Add **Power Platform Import Solution** task to the pipeline.



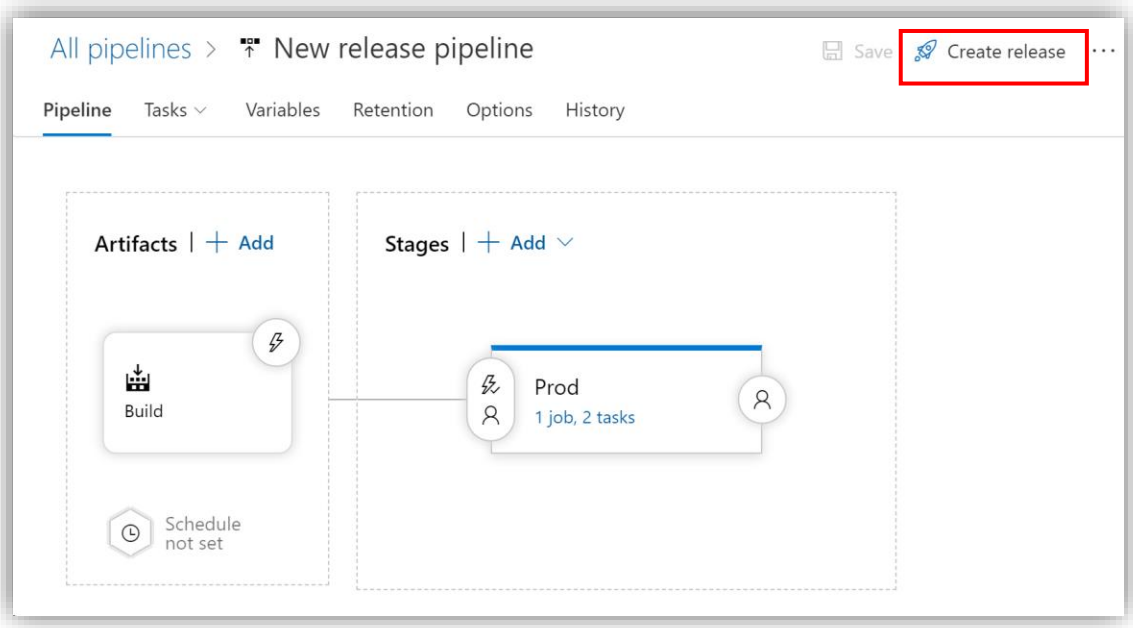
86. Configure the solution import to use the production environment connection and the following for solution input file:
$(System.DefaultWorkingDirectory)/Build/drop/$(SolutionName)_managed.zip
Click **Save**.



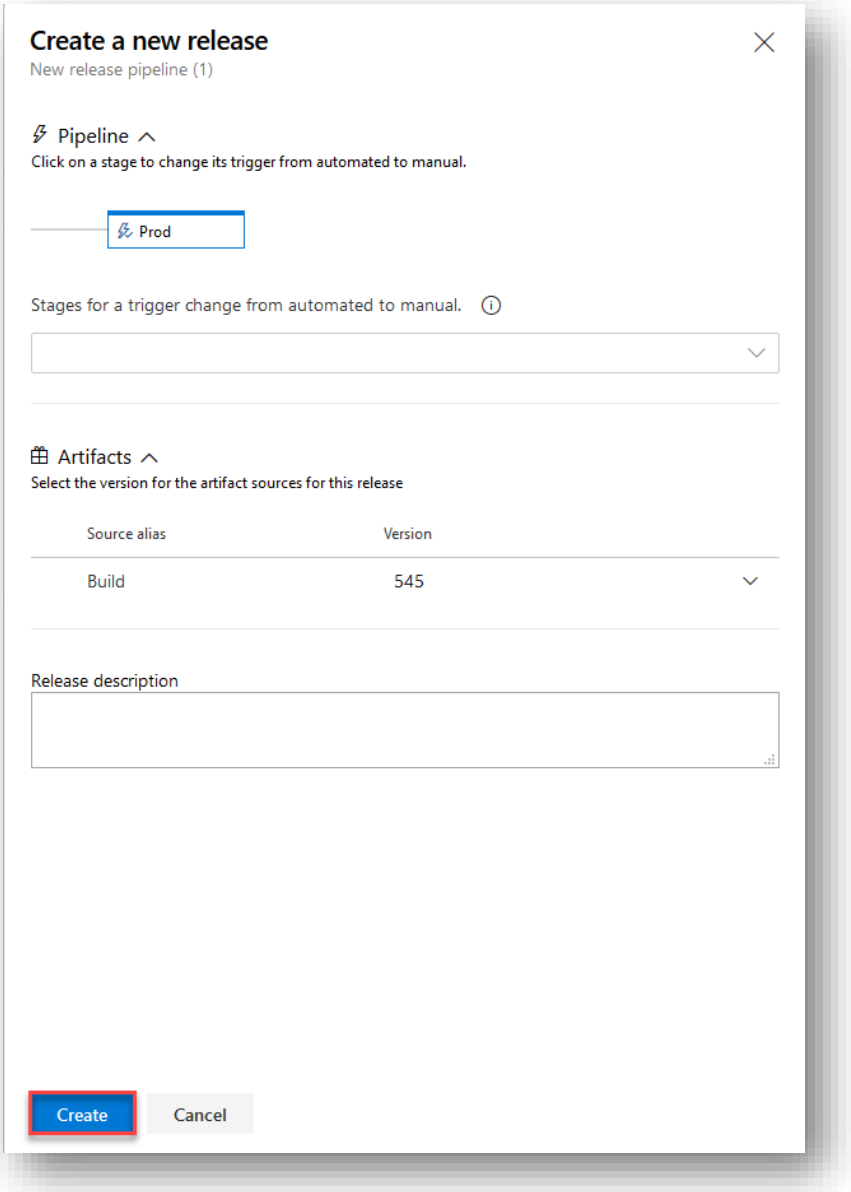
87. Switch to the **Variables** tab and add a variable named “SolutionName” and the unique name of the solution to import. If you are using the solution created in lab 1, the solution name should be “ALMLab”. Click **Save** and then **OK** in the next dialog.



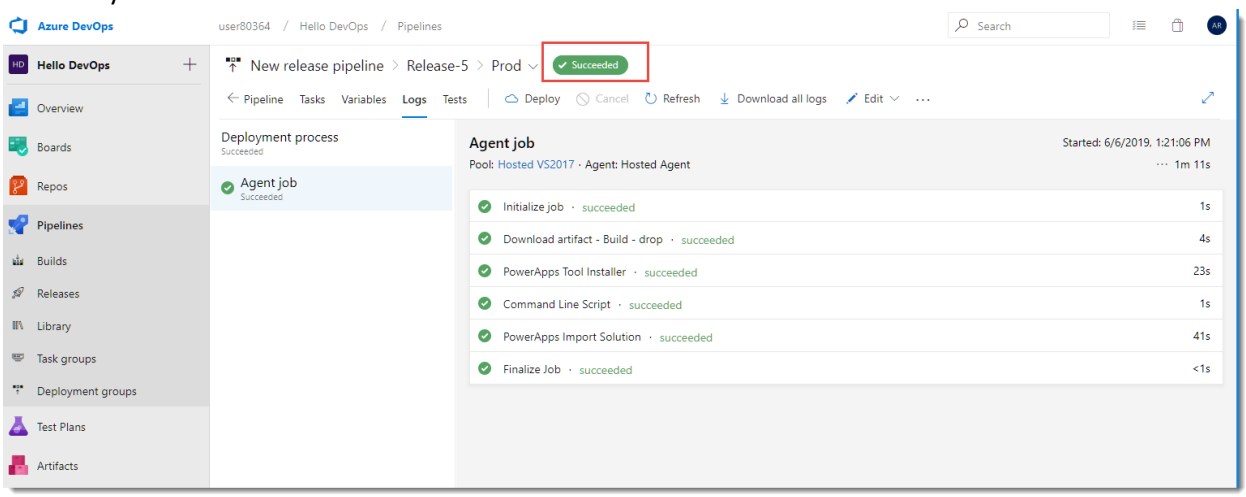
88. Click the **Create release** button. This functions like queuing a build pipeline, only for release pipelines.



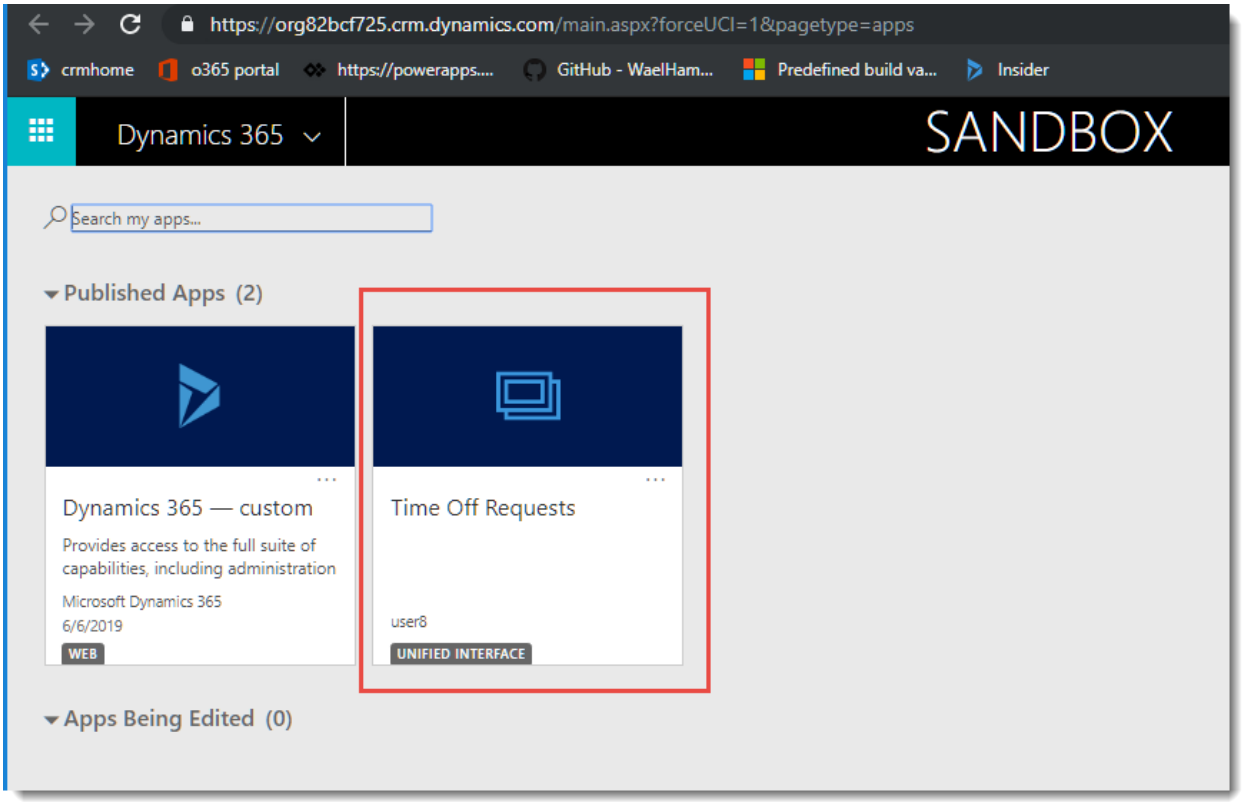
89. Click **Create** on the popup panel.



90. Monitor your release until it is succeeded or troubleshoot if it is not successful.



91. For final confirmation, log into your production system and see your application!



Terms of Use

By using this demo/lab, you agree to the following terms: The technology/functionality described in this demo/lab is provided by Microsoft Corporation for purposes of obtaining your

feedback and to provide you with a learning experience. You may only use the demo/lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this demo/lab or any portion thereof. COPYING OR REPRODUCTION OF THE DEMO/LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED. THIS DEMO/LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS DEMO/LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCTIONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK

If you give feedback about the technology features, functionality and/or concepts described in this demo/lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement. MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE DEMO/LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF DEMO/ LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE DEMO/LAB FOR ANY PURPOSE.

DISCLAIMER

This demo/lab contains only a portion of new features and enhancements in. Some of the features might change in future releases of the product. In this demo/lab, you will learn about some, but not all, new features.