

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

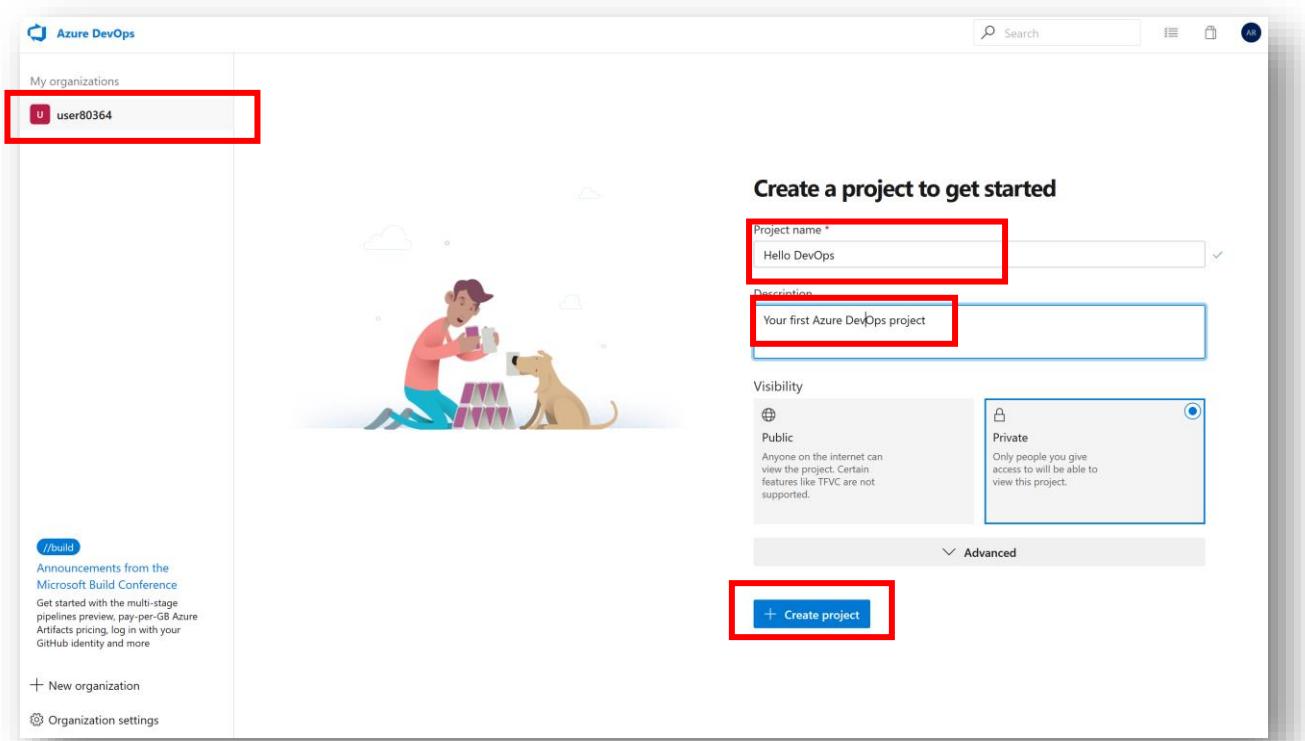
Lab Scenario

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.

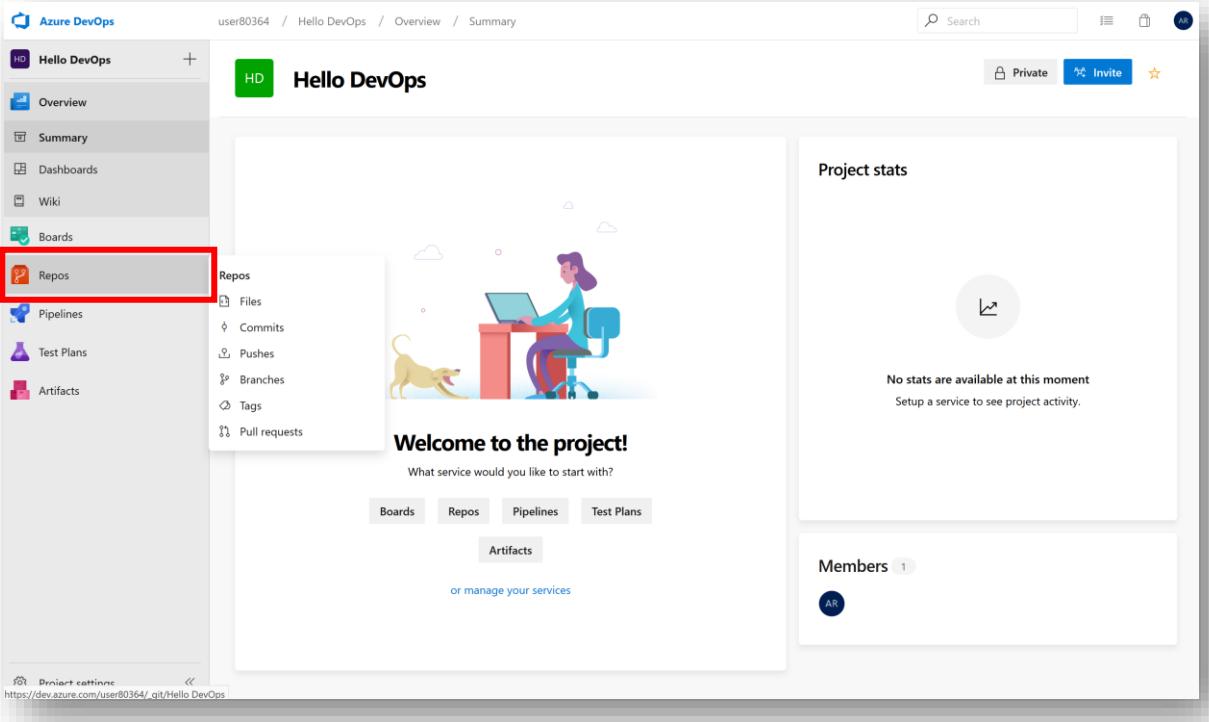
Create an Azure DevOps Project

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**.

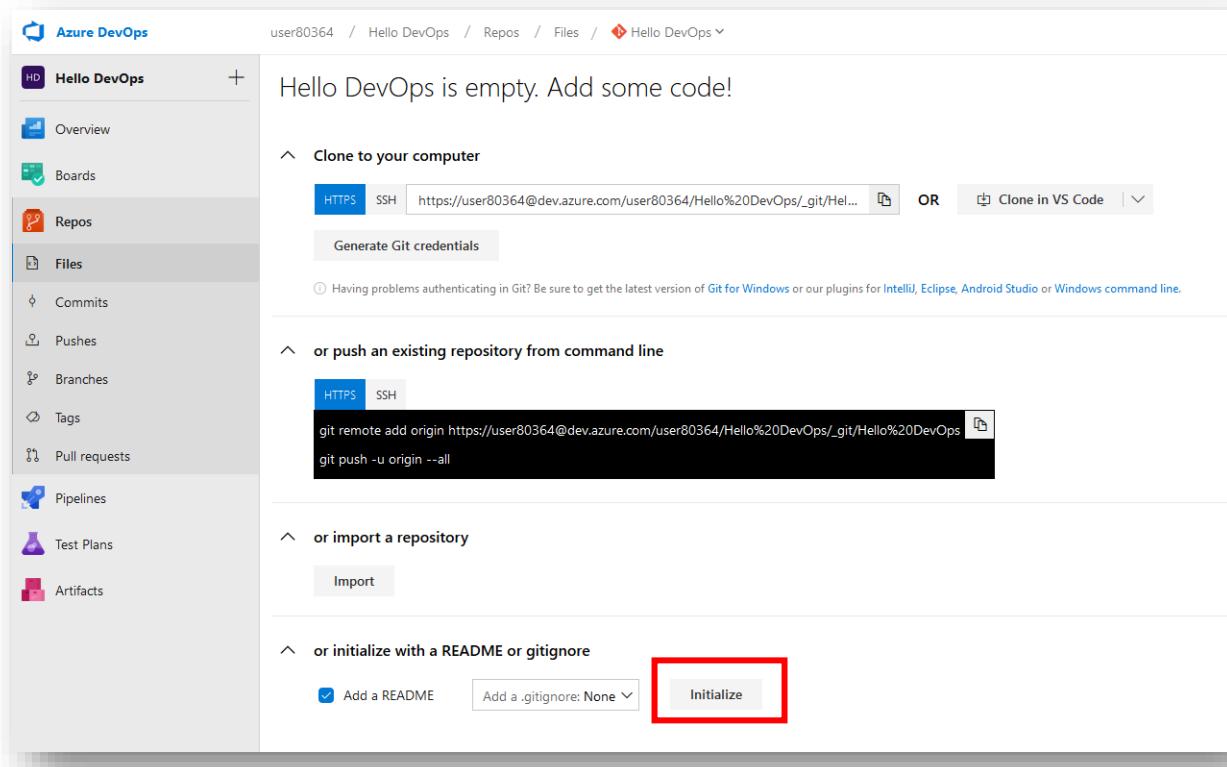


2. When the project is completed, you will need to create a repository to hold the source code.
Click on the **Repos** link in the left navigation.



The screenshot shows the Azure DevOps interface for a project named "Hello DevOps". The left sidebar lists various project services: Overview, Summary, Dashboards, Wiki, Boards, **Repos**, Pipelines, Test Plans, and Artifacts. The "Repos" link is highlighted with a red box. The main content area displays a welcome message "Welcome to the project!" and a "Project stats" section indicating "No stats are available at this moment". Below the stats, there's a note to "Setup a service to see project activity." At the bottom of the main area, there are tabs for Boards, Repos, Pipelines, Test Plans, and Artifacts, with "Repos" being the active tab. A "Members" section shows one member, "AR". The browser address bar at the bottom shows the URL: https://dev.azure.com/user80364/_git>Hello DevOps.

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



The screenshot shows the Azure DevOps interface for a repository named "Hello DevOps". The left sidebar lists various project components like Overview, Boards, Repos, Files, Pipelines, Test Plans, and Artifacts. The main content area displays a message: "Hello DevOps is empty. Add some code!". Below this, there are sections for cloning the repository via HTTPS or SSH, pushing from command line, importing a repository, and initializing it. A red box highlights the "Initialize" button at the bottom right of the initialization section, which includes checkboxes for adding a README and a .gitignore file.

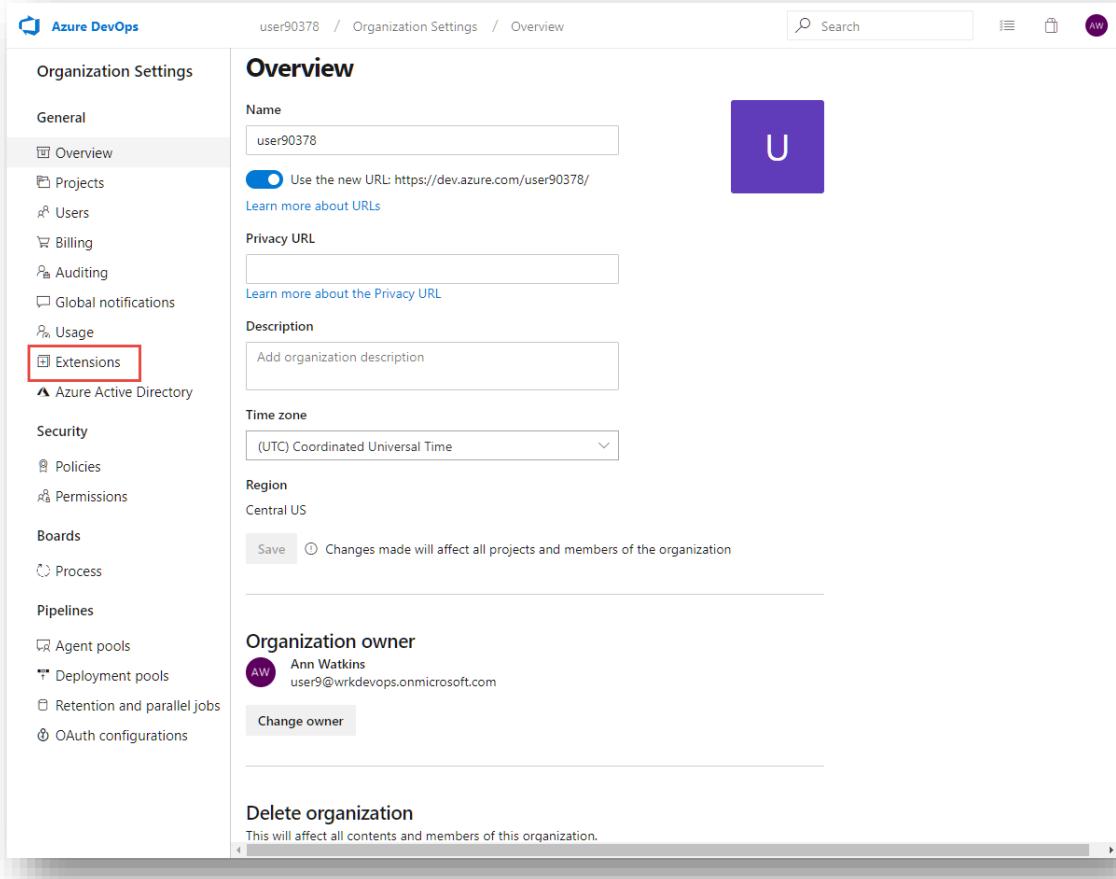


Install and Enable the Azure DevOps Extensions for Power Platform

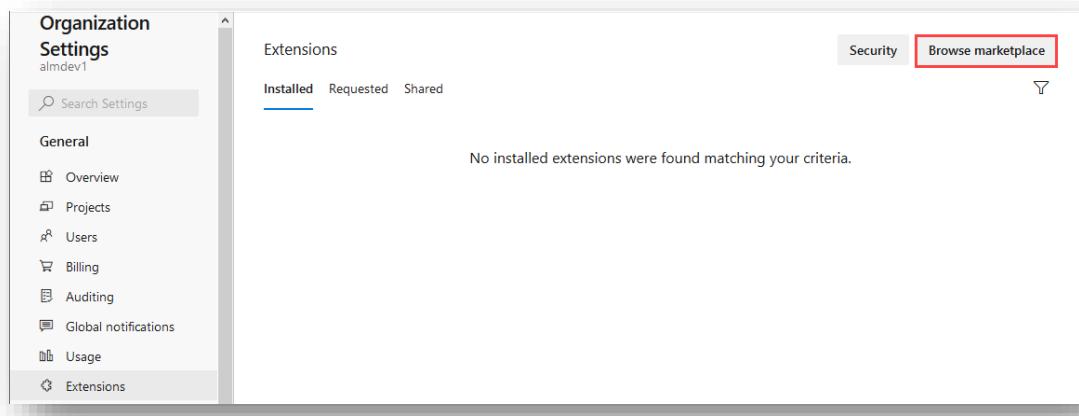
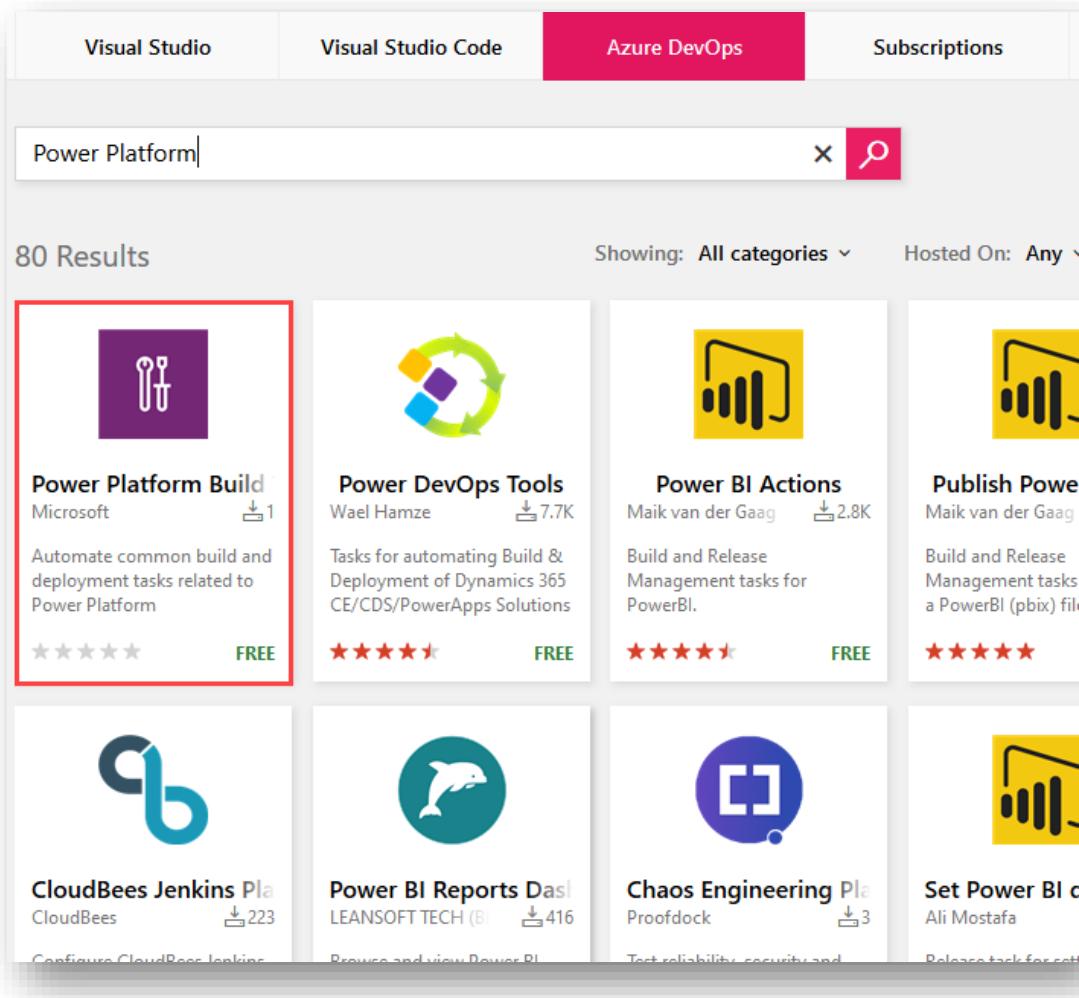
4. On the organization page, click the **Organization settings** link in the bottom left of the navigation panel.

The screenshot shows the Azure DevOps organization page for 'user90378'. The left sidebar includes links for 'My organizations', 'user90378', 'Announcements from the Microsoft Build Conference', 'New organization', and 'Organization settings'. The main area displays a project named 'User9 - Project'. The 'Organization settings' link is highlighted with a red box.

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



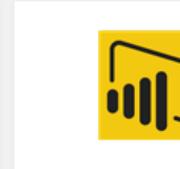
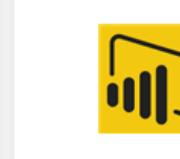
The screenshot shows the 'Organization Settings' page in Azure DevOps. The left sidebar lists various settings categories: General (Overview, Projects, Users, Billing, Auditing, Global notifications, Usage, Extensions), Security (Policies, Permissions), Boards, Process, Pipelines (Agent pools, Deployment pools, Retention and parallel jobs, OAuth configurations). The 'Extensions' link is highlighted with a red box. The main content area is titled 'Overview' and contains fields for Name (user90378), Privacy URL, Description (Add organization description), Time zone (UTC Coordinated Universal Time), and Region (Central US). A 'Save' button and a note about changes affecting all projects and members are present. Below this is a section for 'Organization owner' (Ann Watkins, user9@wrkdevops.onmicrosoft.com) with a 'Change owner' button. At the bottom is a 'Delete organization' section with a warning message: 'This will affect all contents and members of this organization.'

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.

Visual Studio Visual Studio Code **Azure DevOps** Subscriptions

Power Platform | X 🔍

80 Results Showing: All categories ▾ Hosted On: Any ▾

| | | | |
|--|--|---|---|
|  Power Platform Build Microsoft ↓ 1 Automate common build and deployment tasks related to Power Platform ★★★★★ FREE |  Power DevOps Tools Wael Hamze ↓ 7.7K Tasks for automating Build & Deployment of Dynamics 365 CE/CDS/PowerApps Solutions ★★★★★ FREE |  Power BI Actions Maik van der Gaag ↓ 2.8K Build and Release Management tasks for PowerBI. ★★★★★ FREE |  Publish Power BI Maik van der Gaag Build and Release Management tasks for a PowerBI (pbix) file ★★★★★ |
|  CloudBees Jenkins CloudBees ↓ 223 Configure CloudBees Jenkins |  Power BI Reports Dashboard LEANSOFT TECH (B) ↓ 416 Browse and view Power BI |  Chaos Engineering Proofdock ↓ 3 Test reliability, security and |  Set Power BI Ali Mostafa Release task for setting up Power BI |



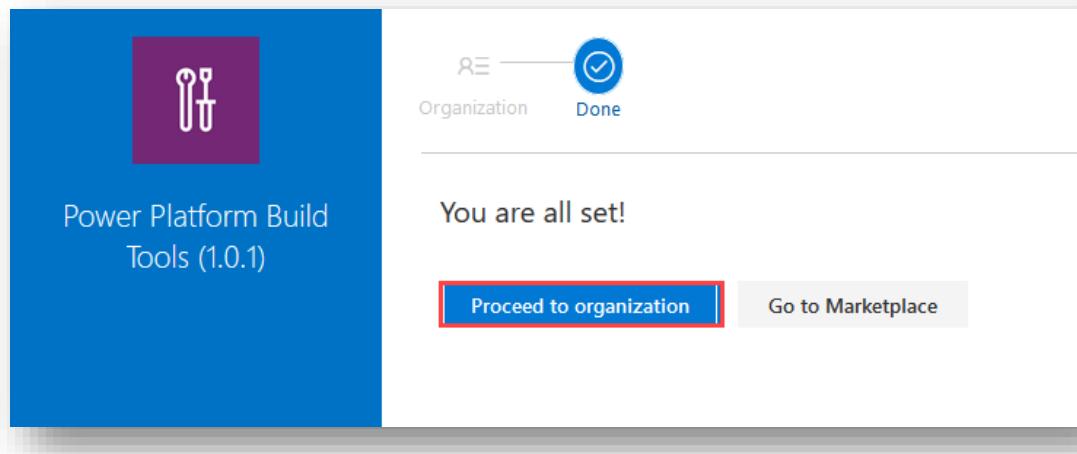
8. Click **Get it free**.

The screenshot shows the Azure DevOps Marketplace page for the "Power Platform Build Tools (1.0.1)" extension. The page includes a purple icon with a wrench and screwdriver, the extension name, a Microsoft badge, a star rating of 0, and a "Free" badge. A green "Get it free" button is highlighted with a red box. Below the main header, there are tabs for "Overview", "Q & A", and "Rating & Review". The "Overview" section contains a brief description of the tool's purpose: automating common build and deployment tasks related to Power Platform. It also links to a "here" button for more information. To the right, a sidebar titled "Agent job 1" lists five components: "Run on agent", "Power Platform Tool Installer", "Power Platform Checker", "Power Platform Export Solution", and "Power Platform Unpack Solution".

9. Click **Install**.

The screenshot shows the Visual Studio Marketplace page for the "Power Platform Build Tools (1.0.1)" extension. The left side features a large blue background area with the extension's icon and name. The right side has a white background with a "Done" progress bar, a dropdown menu set to "almdev1", and a prominent red "Install" button. Below these, there is a note about "For Azure DevOps Server" and a "Download" link.

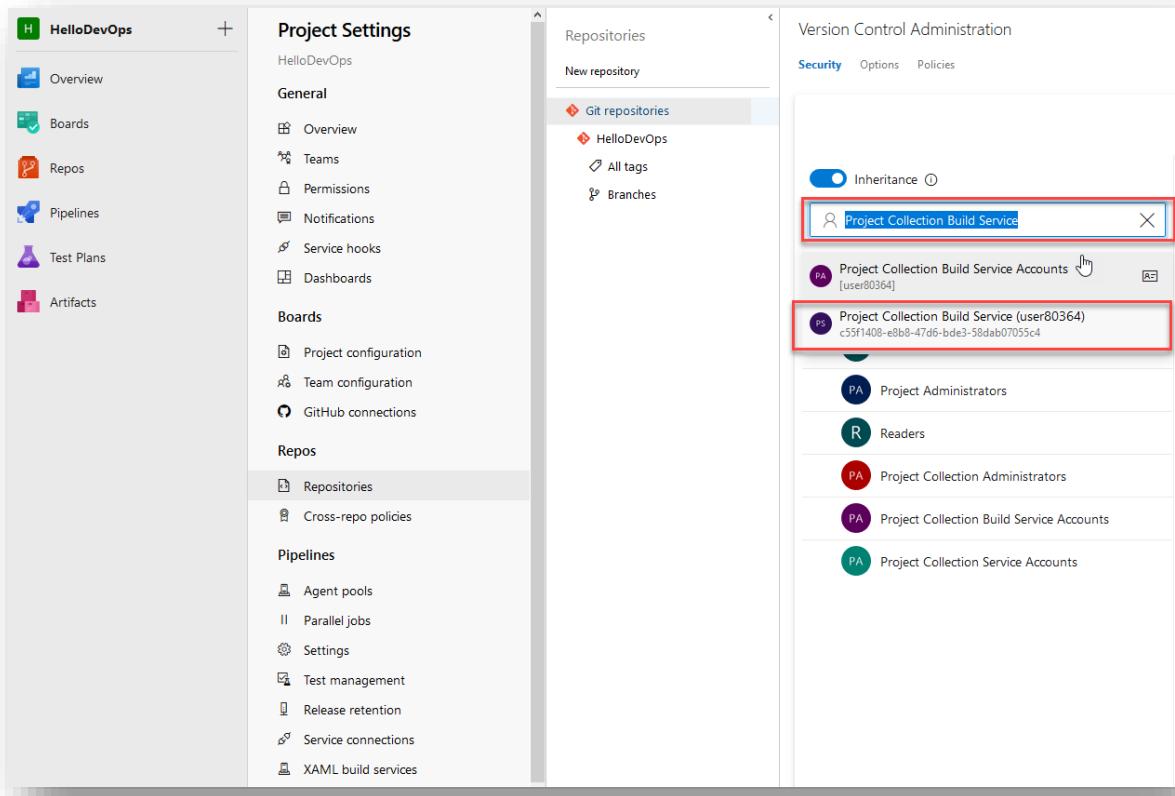
10. Click **Proceed to organization**.



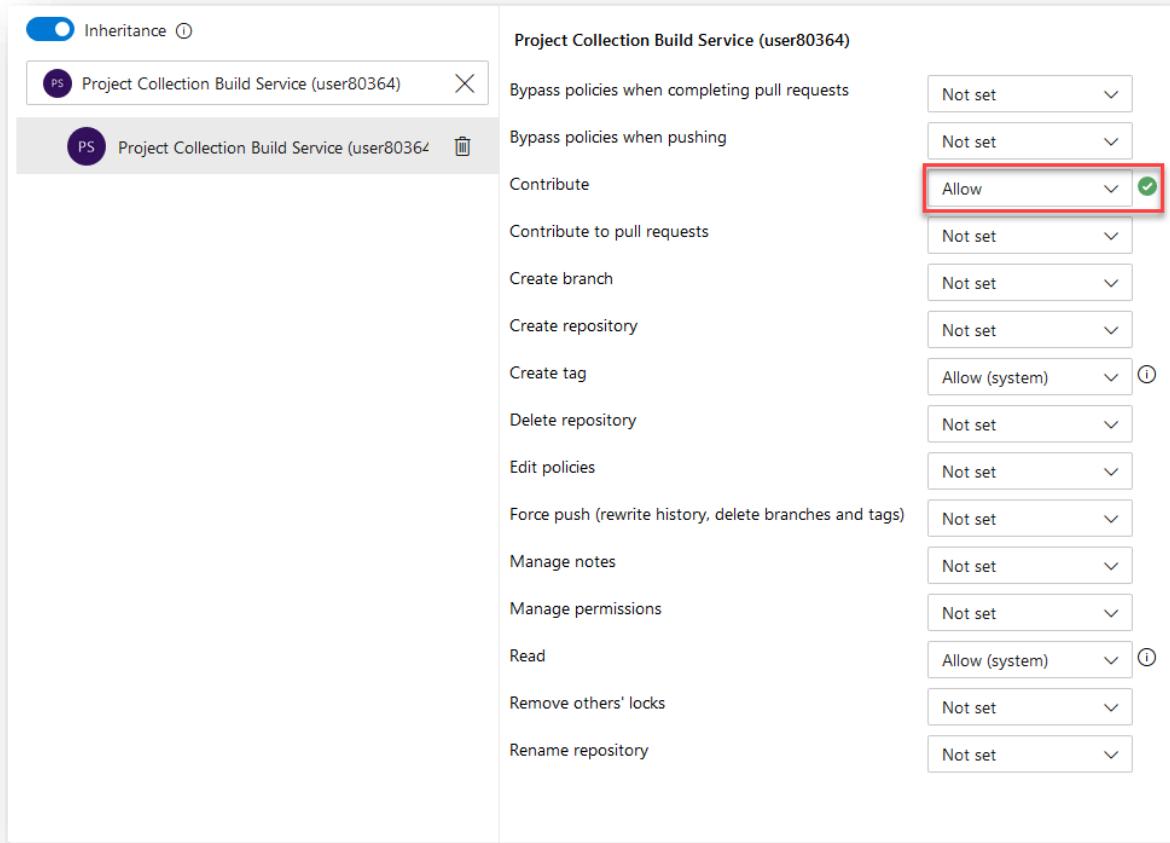
Configure Azure DevOps Permissions for Build Service Account

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.

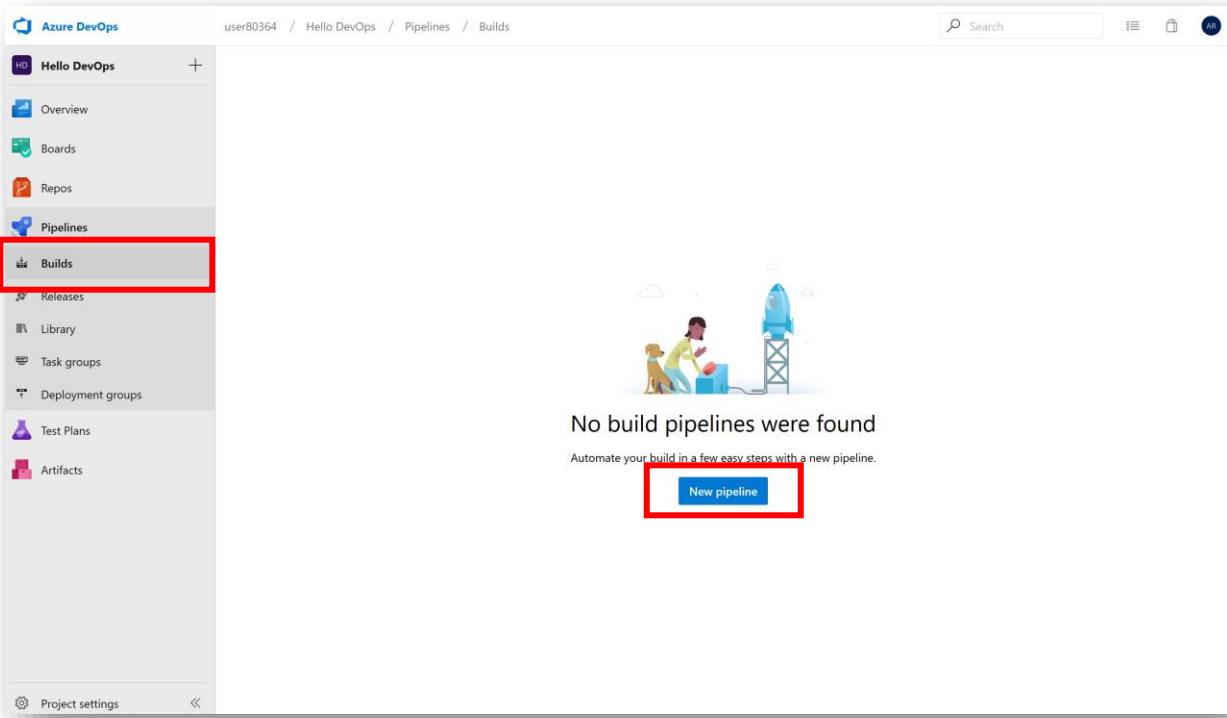


The screenshot shows the 'Project Collection Build Service (user80364)' settings page. The 'Inheritance' toggle is turned on. A user named 'Project Collection Build Service (user80364)' is selected. The 'Contribute' permission is highlighted with a red box and has a green checkmark indicating it is set to 'Allow'. Other permissions listed include 'Bypass policies when completing pull requests', 'Bypass policies when pushing', 'Create branch', 'Create repository', 'Create tag', 'Delete repository', 'Edit policies', 'Force push (rewrite history, delete branches and tags)', 'Manage notes', 'Manage permissions', 'Read', 'Remove others' locks', and 'Rename repository'. Most of these permissions are set to 'Not set' or 'Allow (system)'.

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.

Azure DevOps

user80364 / Hello DevOps / Pipelines

Hello DevOps + Connect Select Configure Review

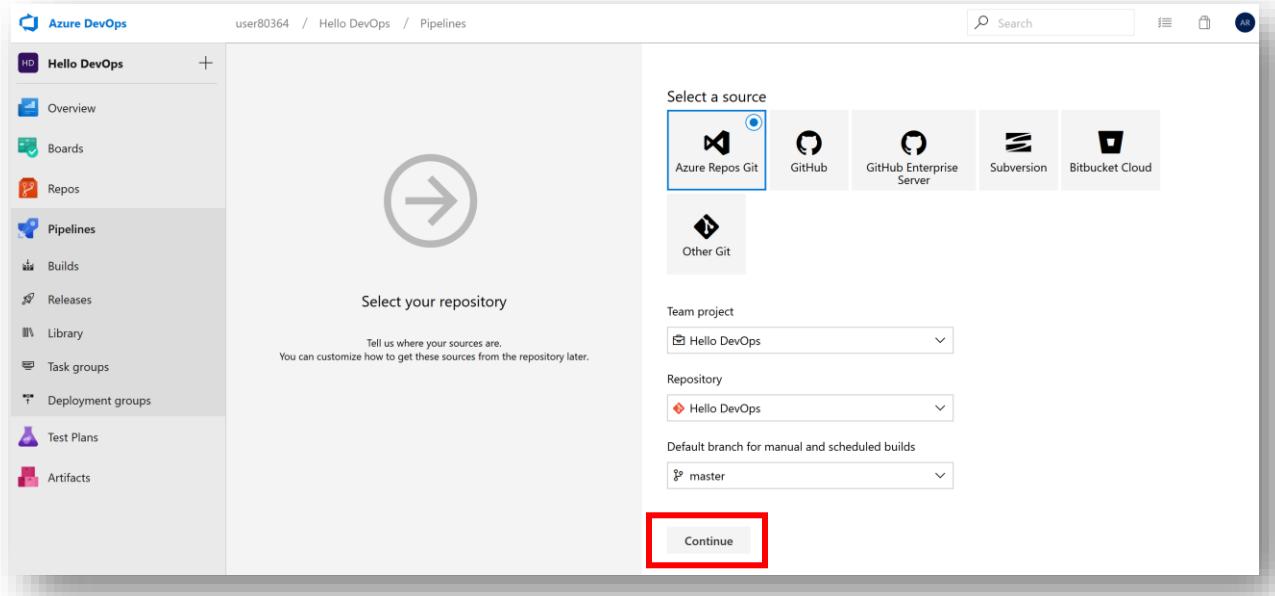
New pipeline

Where is your code?

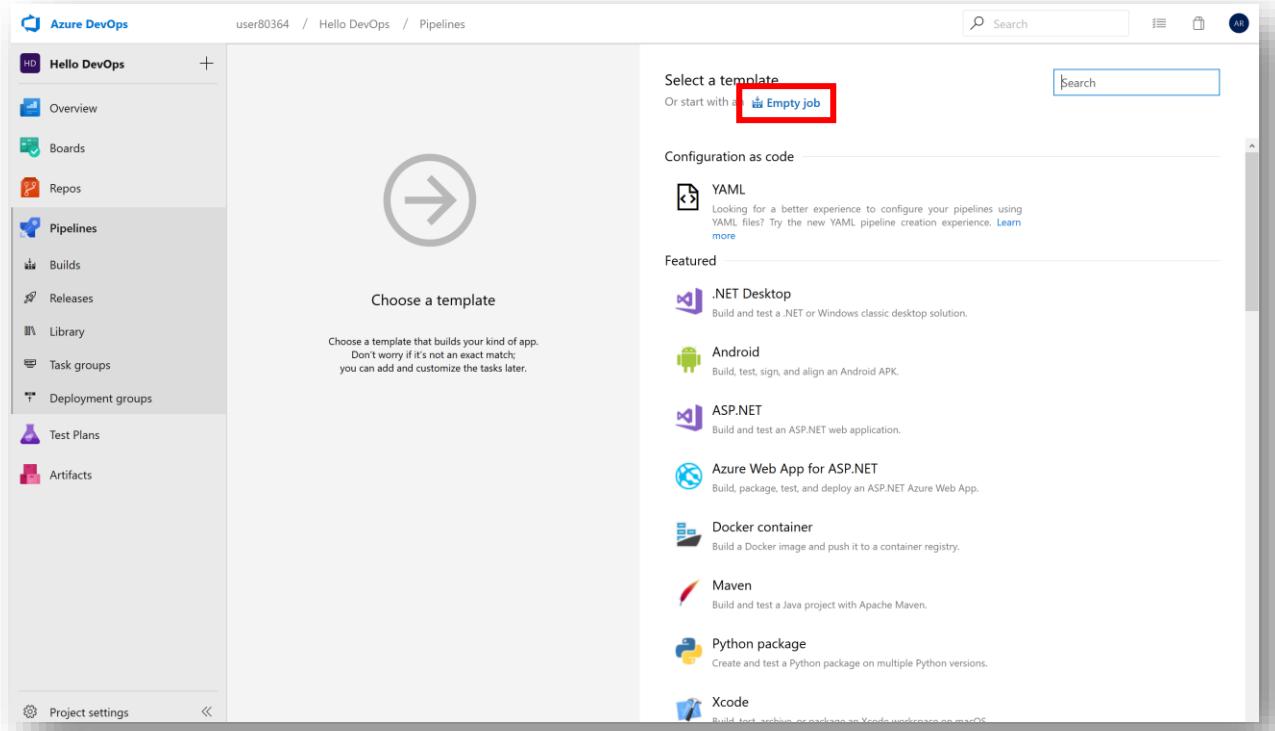
- Azure Repos Git YAML
Free private Git repositories, pull requests, and code search
- Bitbucket Cloud YAML
Hosted by Atlassian
- Github YAML
Home to the world's largest community of developers
- Github Enterprise Server YAML
The self-hosted version of GitHub Enterprise
- Other Git
Any Internet-facing Git repository
- Subversion
Centralized version control by Apache

[Use the classic editor](#) to create a pipeline without YAML.

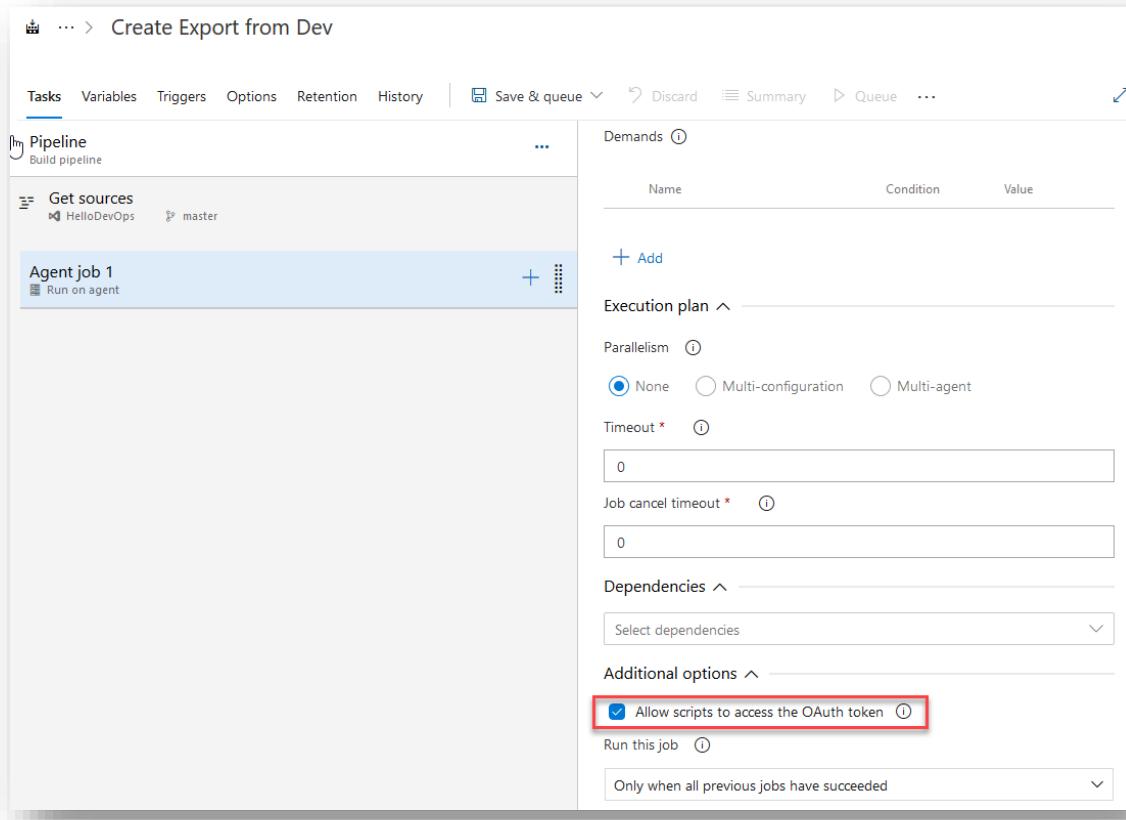
15. Leave default values as is and click **Continue**.



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

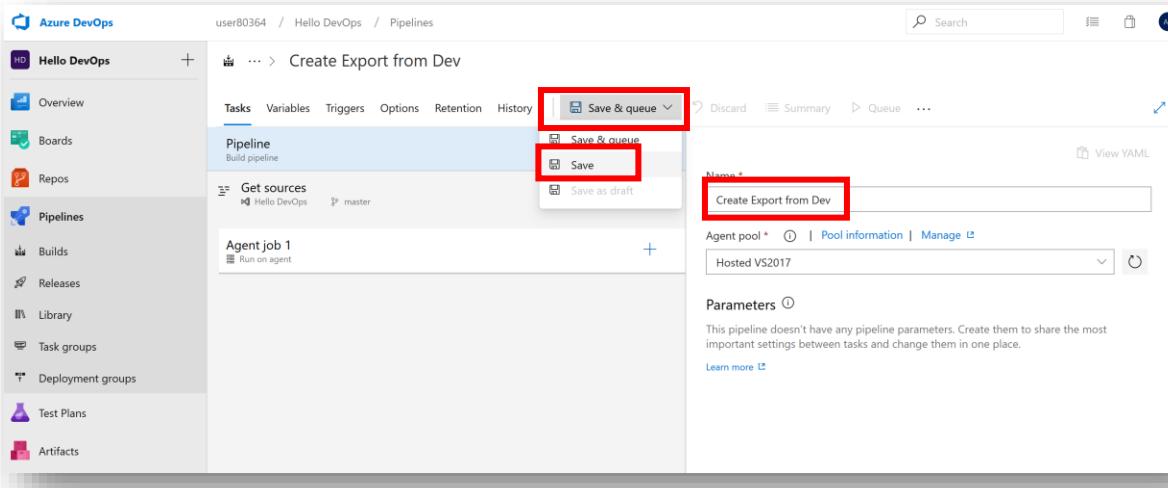


17. Select **Agent job 1** and enable the **Allow scripts to access the OAuth token** option.



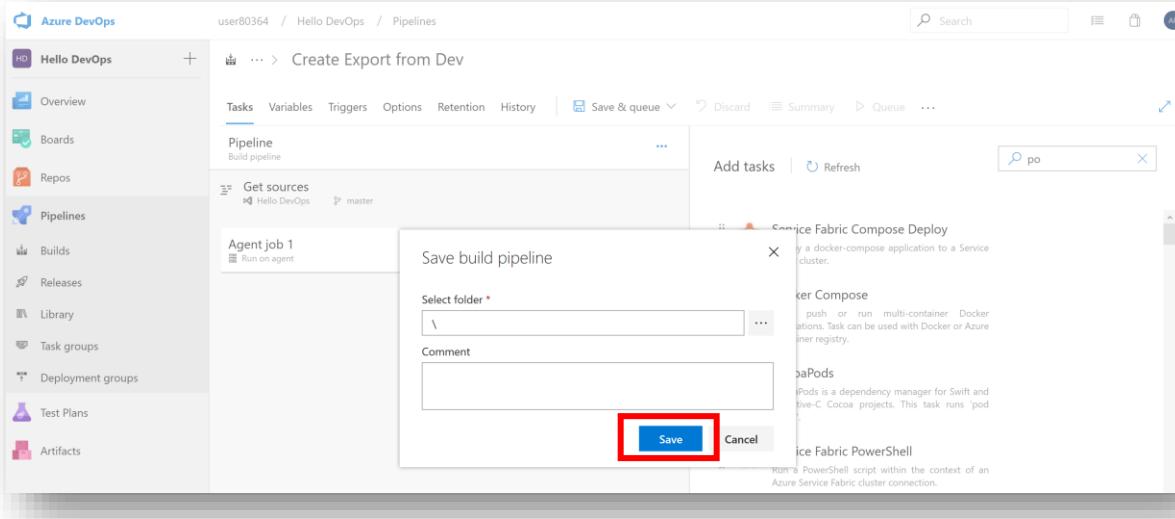
The screenshot shows the Azure DevOps Pipeline editor for the 'Create Export from Dev' pipeline. The 'Agent job 1' step is selected. In the 'Additional options' section, the 'Allow scripts to access the OAuth token' checkbox is checked and highlighted with a red box.

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

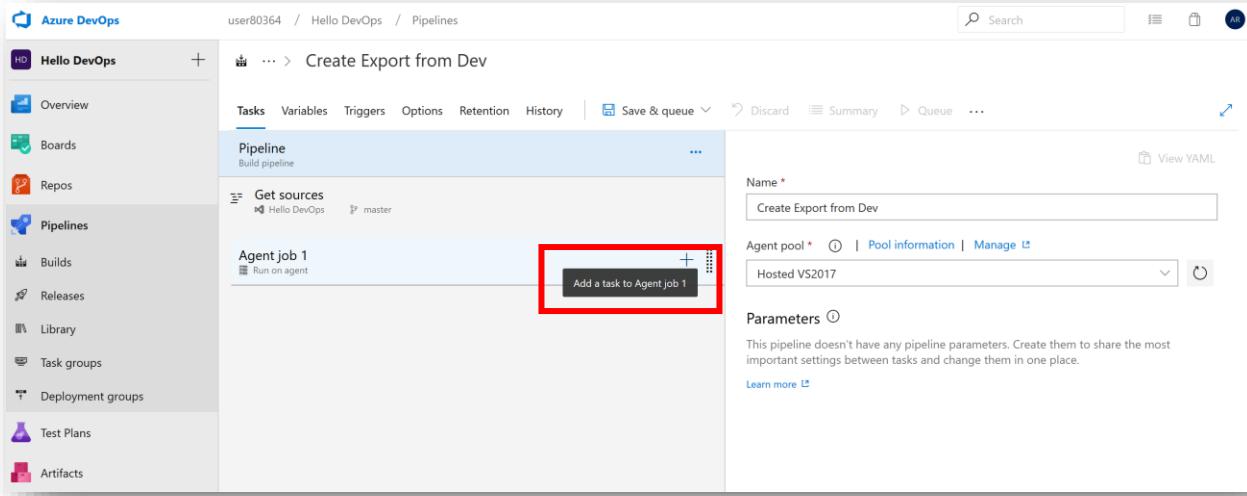


The screenshot shows the Azure DevOps Pipelines page for the 'Create Export from Dev' pipeline. The 'Save & queue' button is highlighted with a red box. The 'Name' field contains 'Create Export from Dev'.

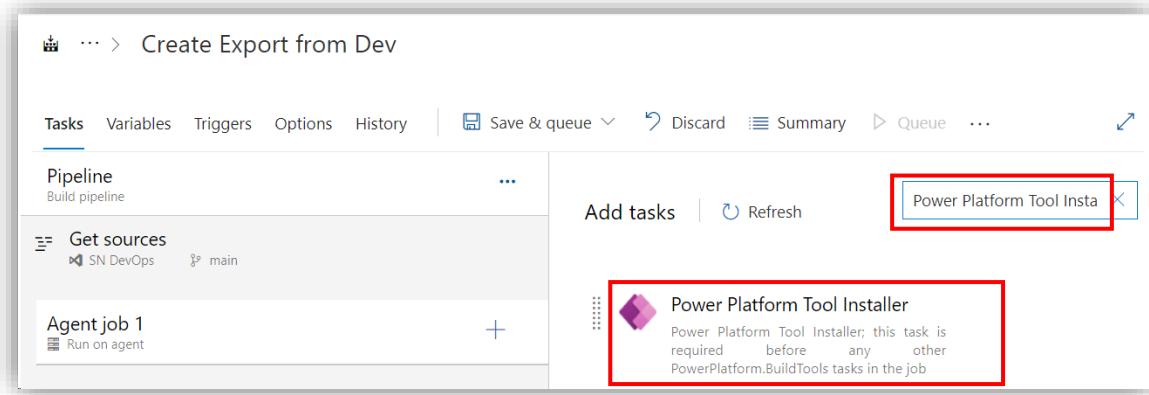
19. Changes to your pipeline are also checked into source control, so select the folder, type a comment, and click **Save**.



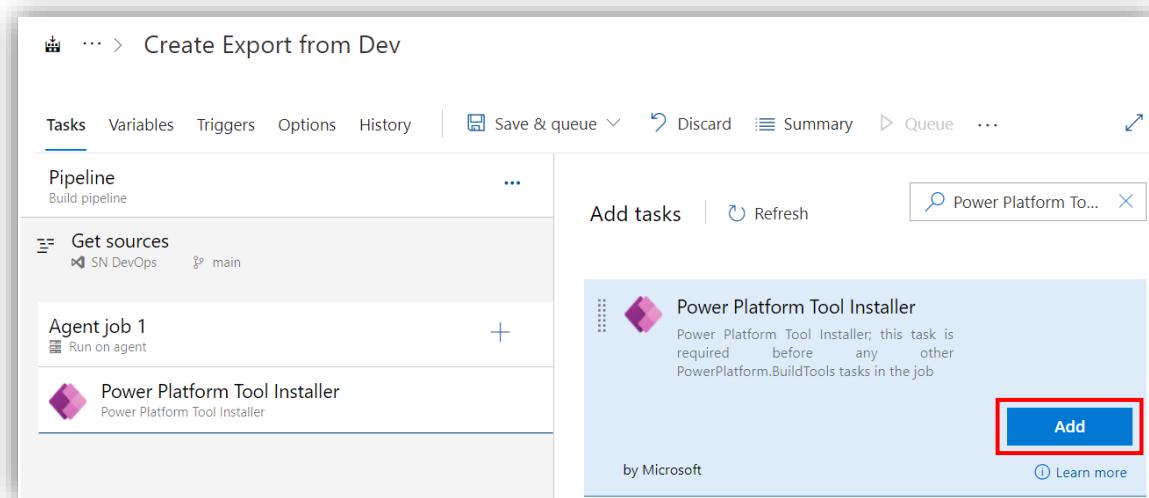
20. You are now ready to start using the Power Platform Build Tools tasks for Azure DevOps in your pipeline. Select **Add a task to Agent job 1**.



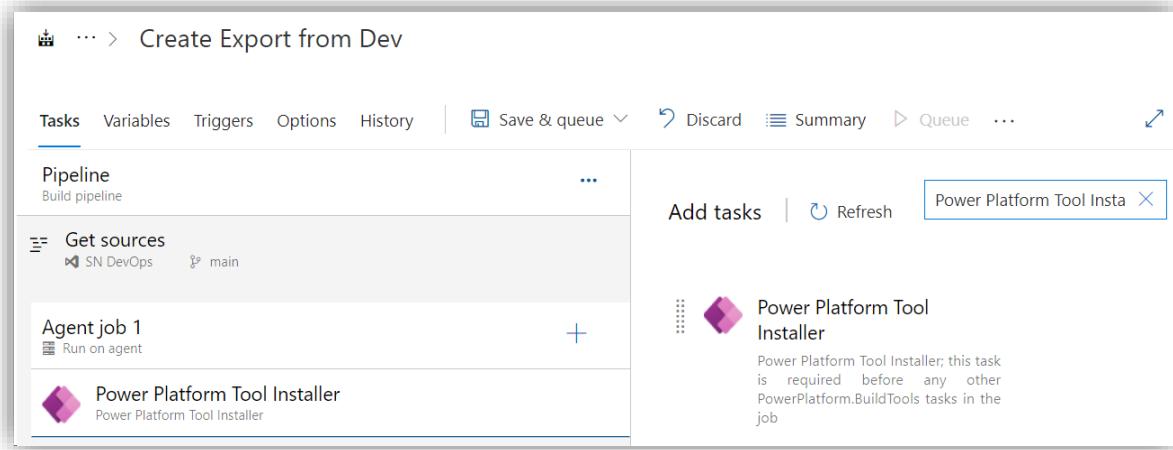
21. The very first task you will perform is one that installs the required tools on the agent. Search for “Power Platform Tool”.



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.

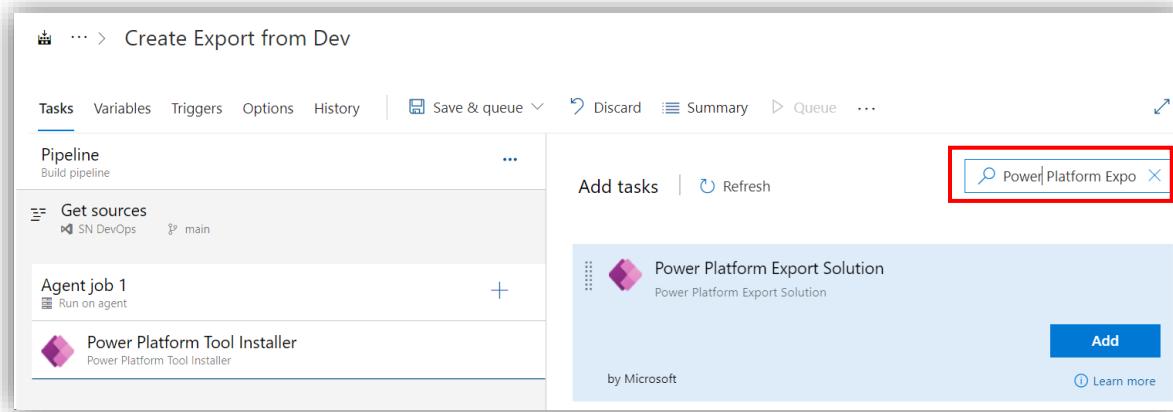


The screenshot shows the 'Create Export from Dev' pipeline in Azure DevOps. The pipeline consists of a 'Get sources' step and an 'Agent job 1' step. Within 'Agent job 1', there is a 'Power Platform Tool Installer' task. On the right side of the screen, a search bar is highlighted with the text 'Power Platform Tool Insta'. A tooltip for the 'Power Platform Tool Installer' task is displayed, stating: 'Power Platform Tool Installer; this task is required before any other PowerPlatform.BuildTools tasks in the job.'

Export Solution as Unmanaged

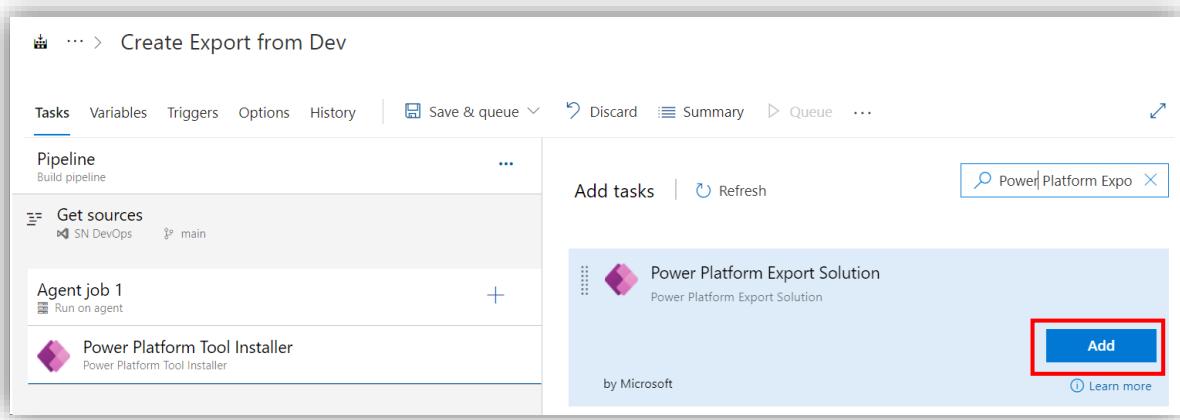
Your unmanaged solution is your source code for your configuration. You should export your solution as unmanaged to check it into source control.

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

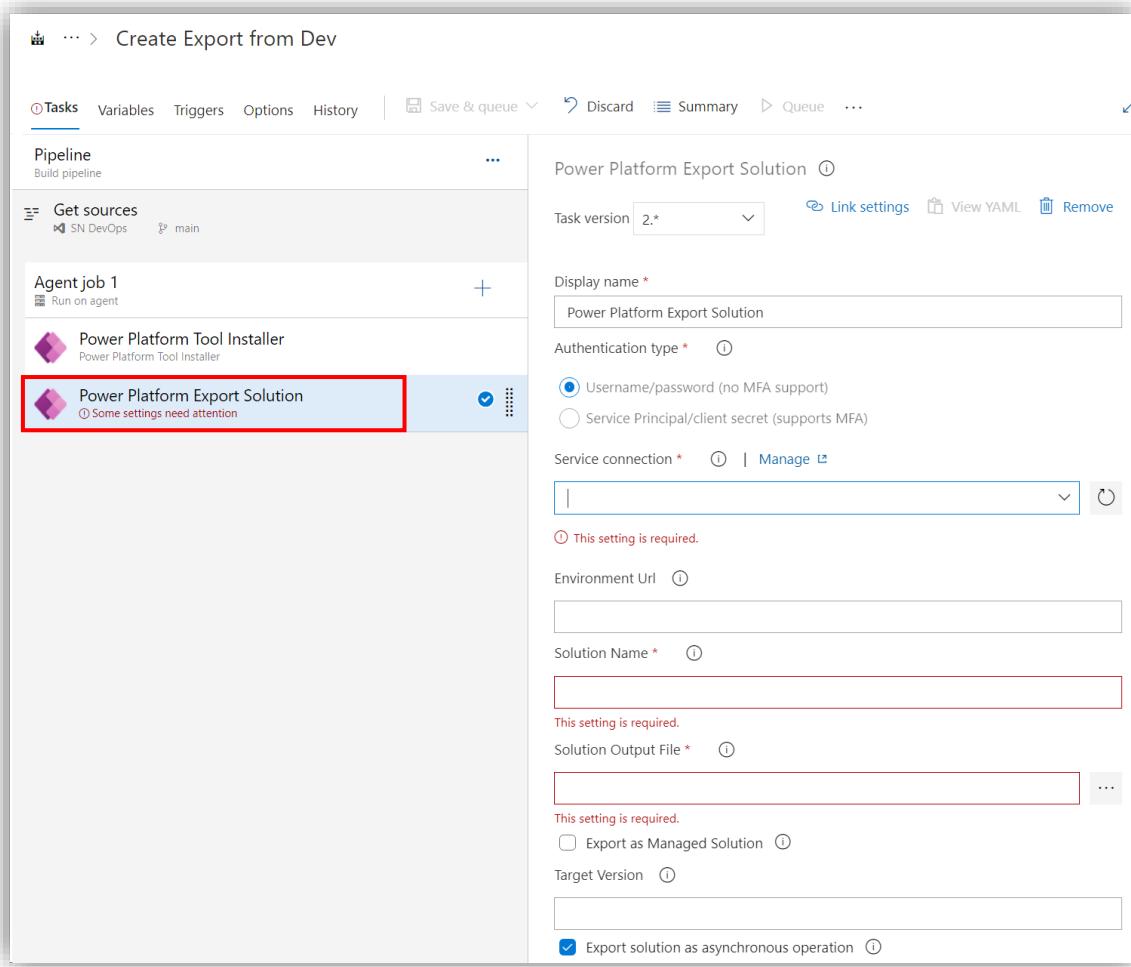


The screenshot shows the same pipeline interface as above. The search bar on the right is now highlighted with the text 'Power Platform Expo'. A tooltip for the 'Power Platform Export Solution' task is displayed, stating: 'Power Platform Export Solution by Microsoft'.

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



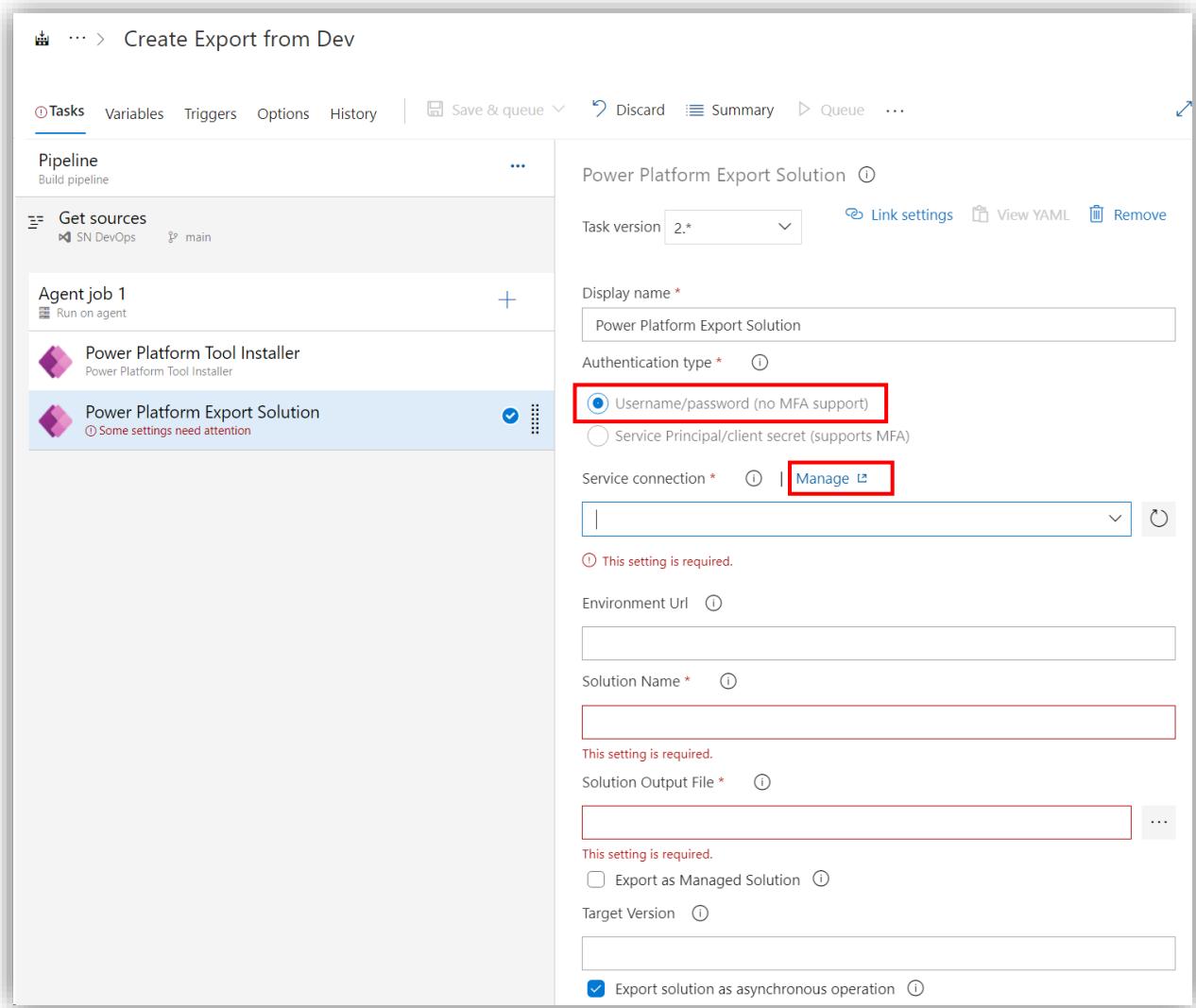
26. After adding the Power Platform Export Solution task, you will notice that additional configuration is required. Click on **Power Platform Export Solution** in the pipeline view.



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**.



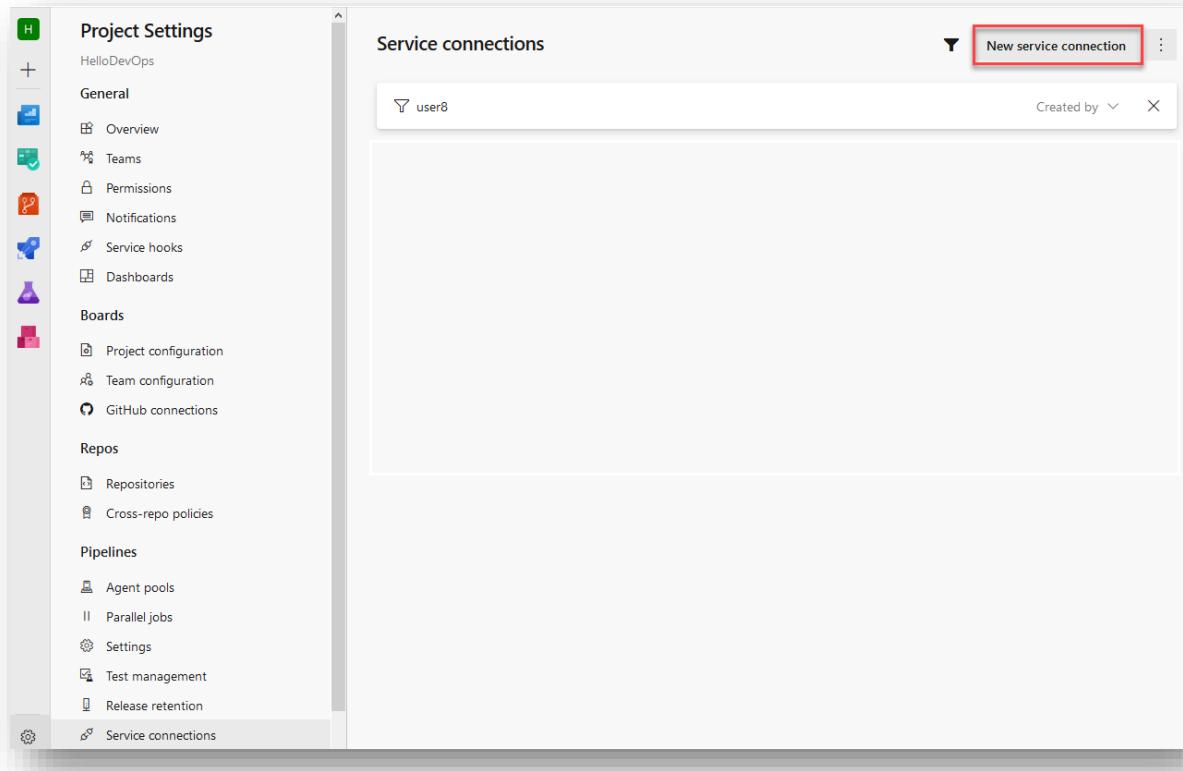
The screenshot shows the 'Create Export from Dev' pipeline configuration in the Power Platform Studio. The pipeline consists of the following steps:

- Get sources**: SN DevOps, main
- Agent job 1**: Run on agent
- Power Platform Tool Installer**: Power Platform Tool Installer
- Power Platform Export Solution**: Some settings need attention

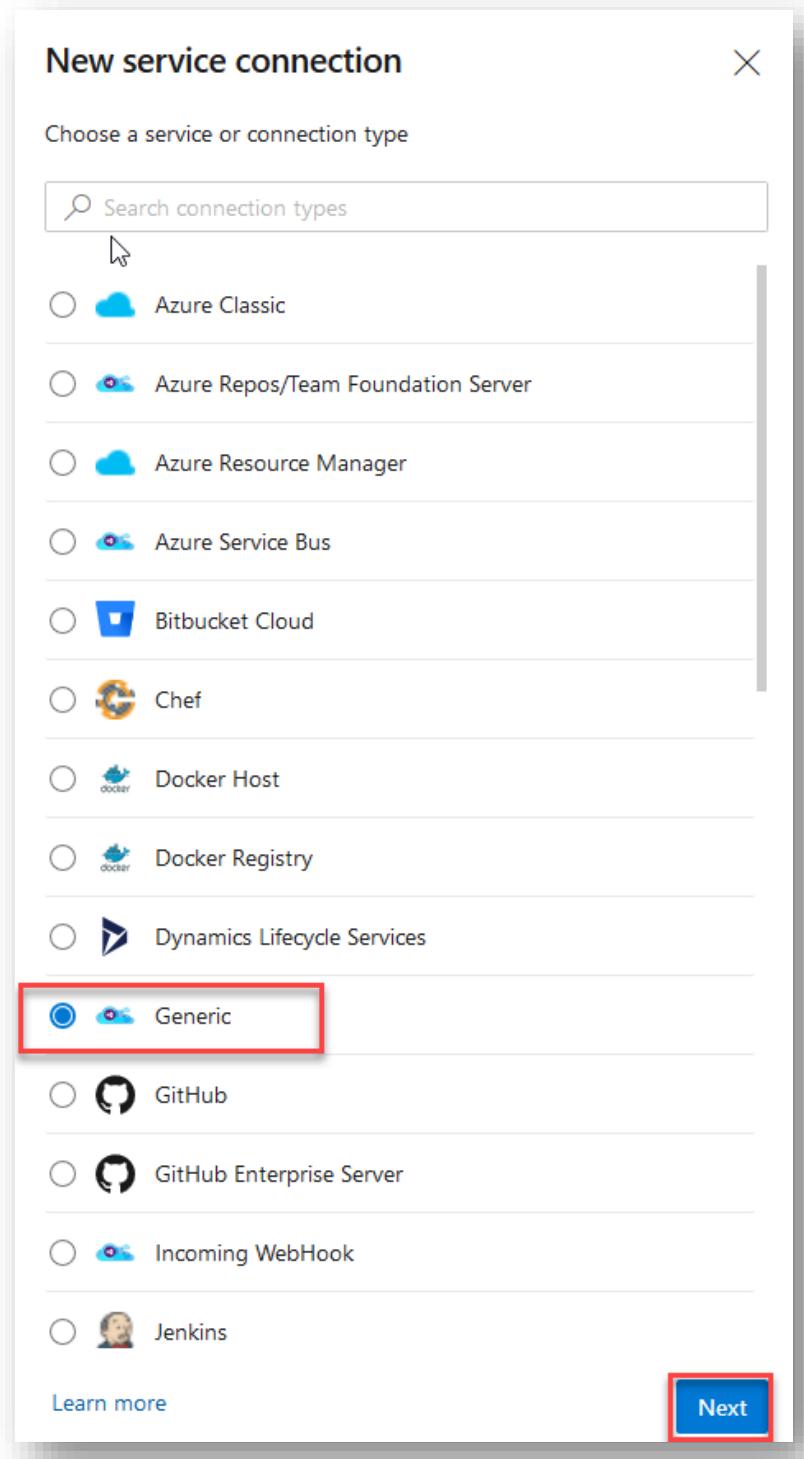
For the final step, 'Power Platform Export Solution', the configuration details are as follows:

- Display name**: Power Platform Export Solution
- Authentication type**: Username/password (no MFA support) (highlighted with a red box)
- Service connection**: | **Manage** (highlighted with a red box)
- Environment Url**:
- Solution Name**:
- Solution Output File**:
- Target Version**:
- Export solution as asynchronous operation**:

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.

New Generic service connection X

Server URL

Authentication

Username (optional) Password/Token Key (optional)
Username for connecting to the endpoint Password/Token Key for connecting to the endpoint

Details

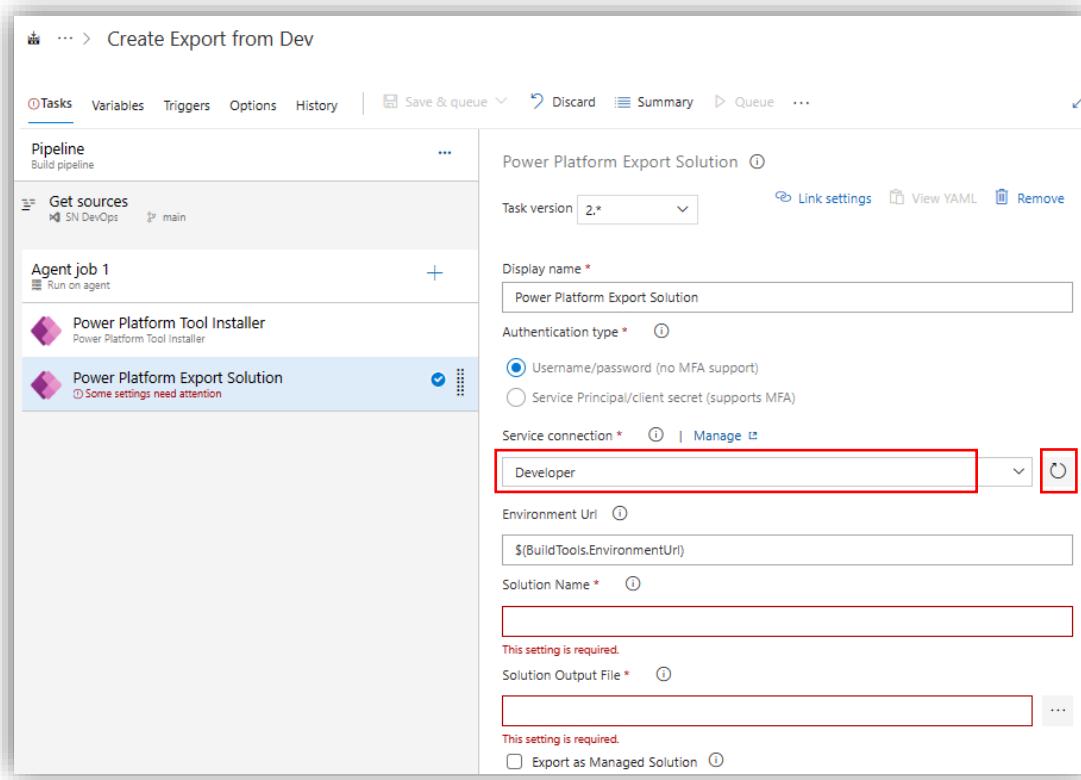
Service connection name

Description (optional)

Security
 Grant access permission to all pipelines

[Learn more](#) [Troubleshoot](#) [Back](#) [Save](#)

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.



The screenshot shows the Azure DevOps Pipeline Editor for a pipeline named "Create Export from Dev". The pipeline consists of three tasks:

- "Get sources" (SN DevOps) - main
- "Agent job 1" (Run on agent)
 - "Power Platform Tool Installer"
 - "Power Platform Export Solution" (Some settings need attention)

The "Power Platform Export Solution" task is currently selected. Its configuration pane is displayed on the right:

- Display name:** Power Platform Export Solution
- Authentication type:** Username/password (no MFA support) (selected)
- Service connection:** Developer (highlighted with a red box)
- Environment Url:** \$(BuildTools.EnvironmentUrl)
- Solution Name:** (This setting is required)
- Solution Output File:** (This setting is required)
 - Export as Managed Solution

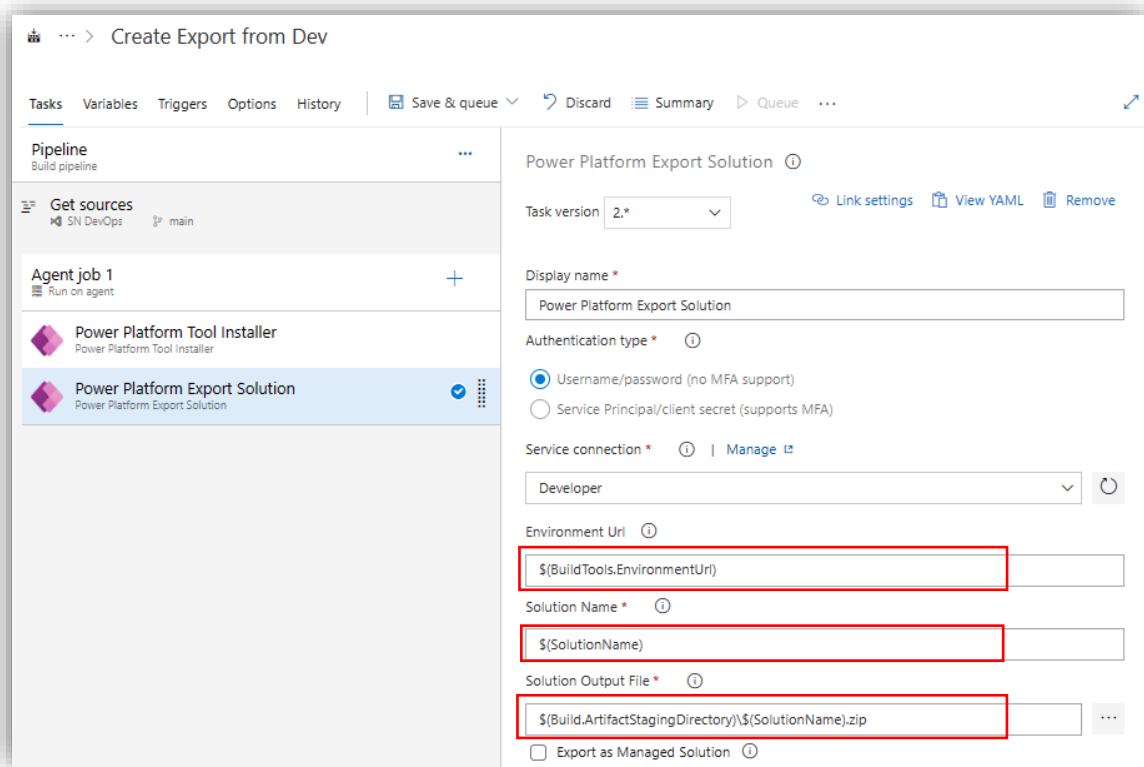
34. Fill out the remaining details:

- 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.

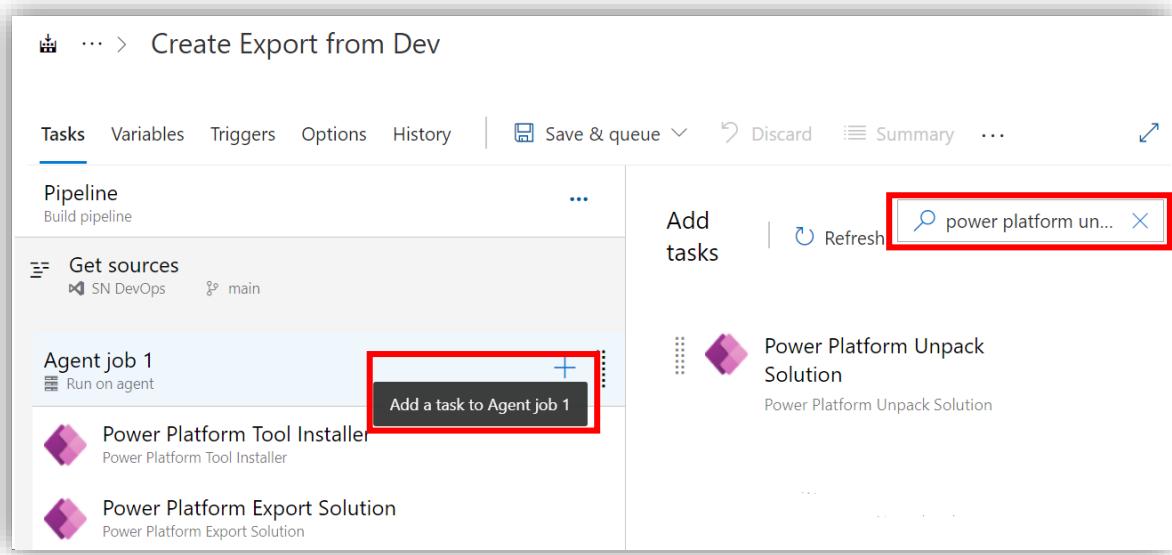


35. Save the build pipeline by clicking **Save & queue**, then save from the top command bar and clicking **Save** on the dialog.

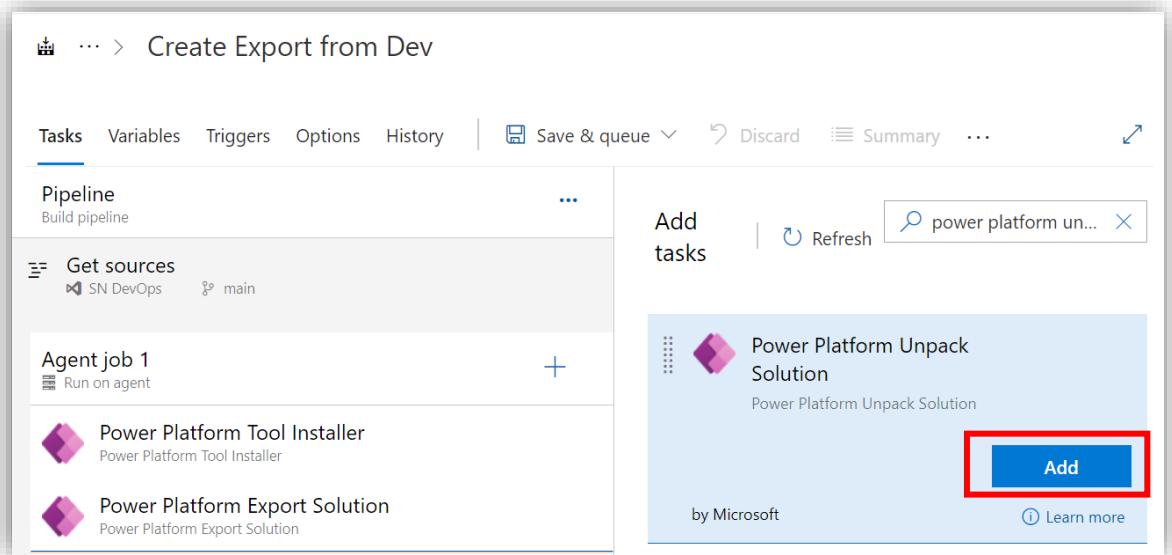
Unpack Solution

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**.

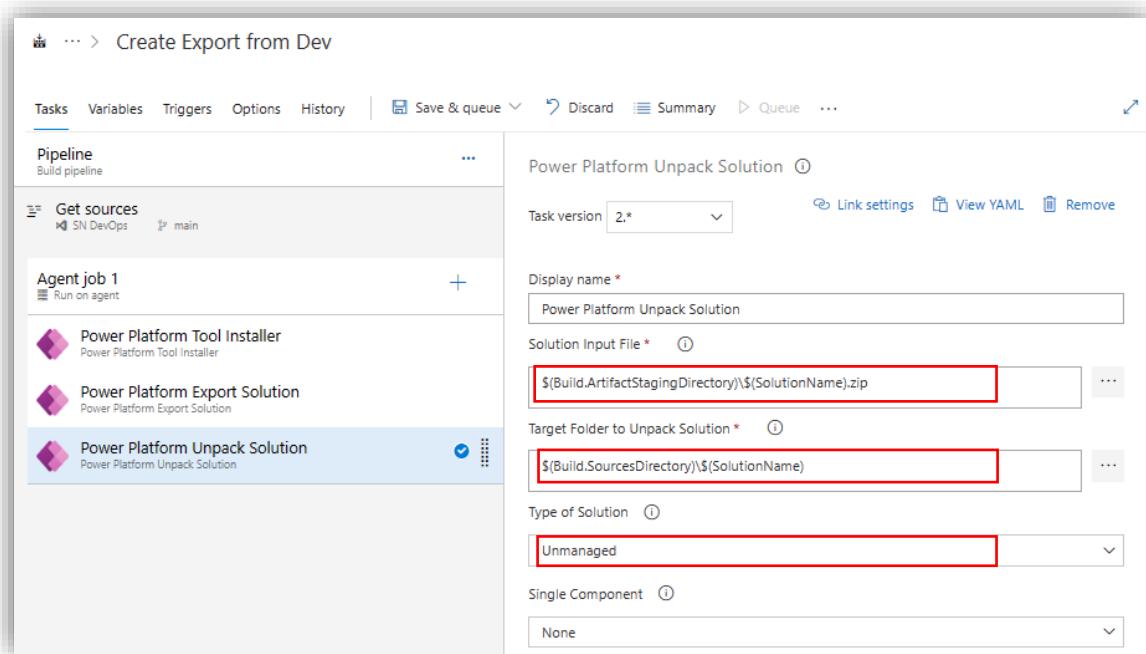


37. Add the **Power Platform Unpack Solution** task to the pipeline.

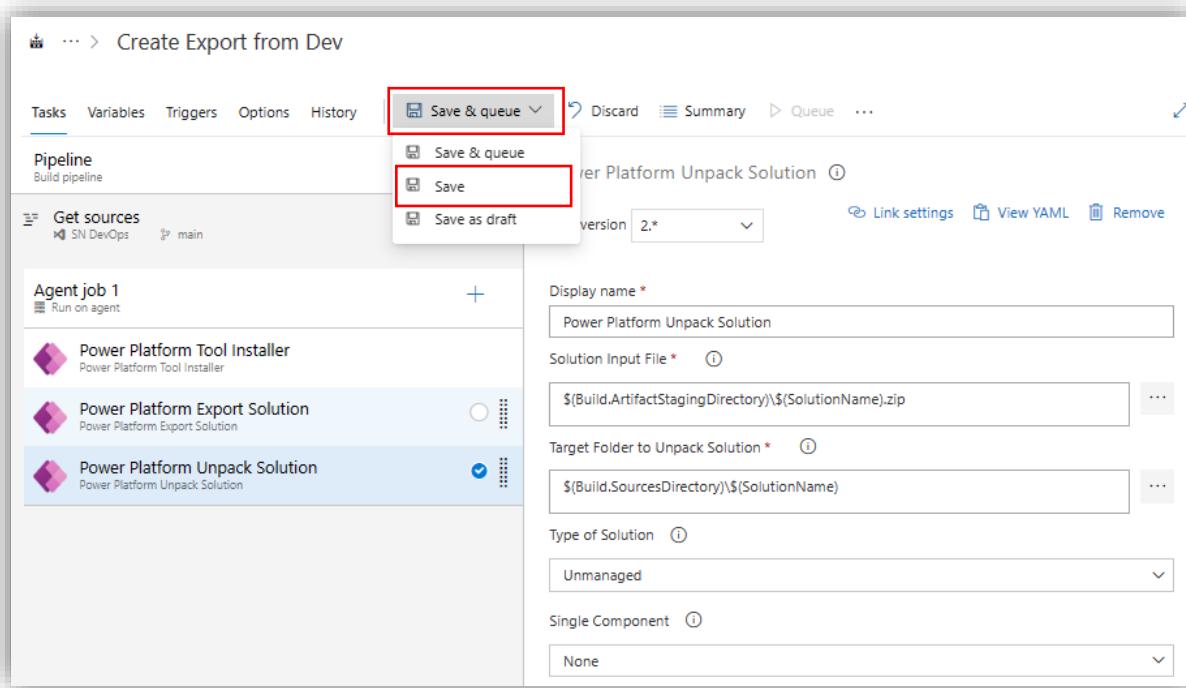


38. Open the task to configure settings for the task with the following details:

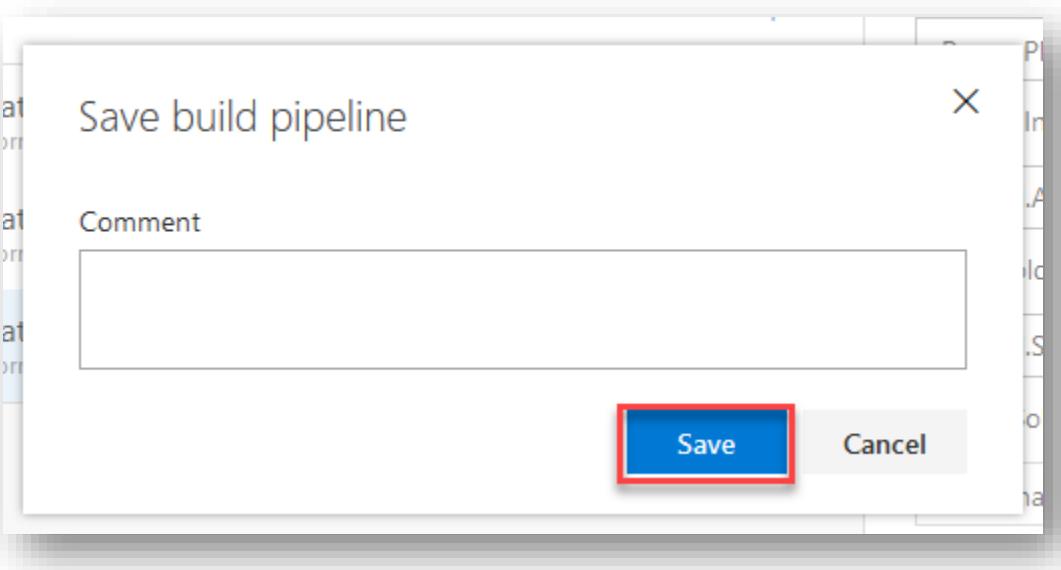
- Solution Input File: `$(Build.ArtifactStagingDirectory)\$(SolutionName).zip`
- Target Folder to Unpack Solution: `$(Build.SourcesDirectory)\$(SolutionName)`
- Type of solution: **Unmanaged**



39. Save the updated pipeline.



40. You can leave the comment blank.

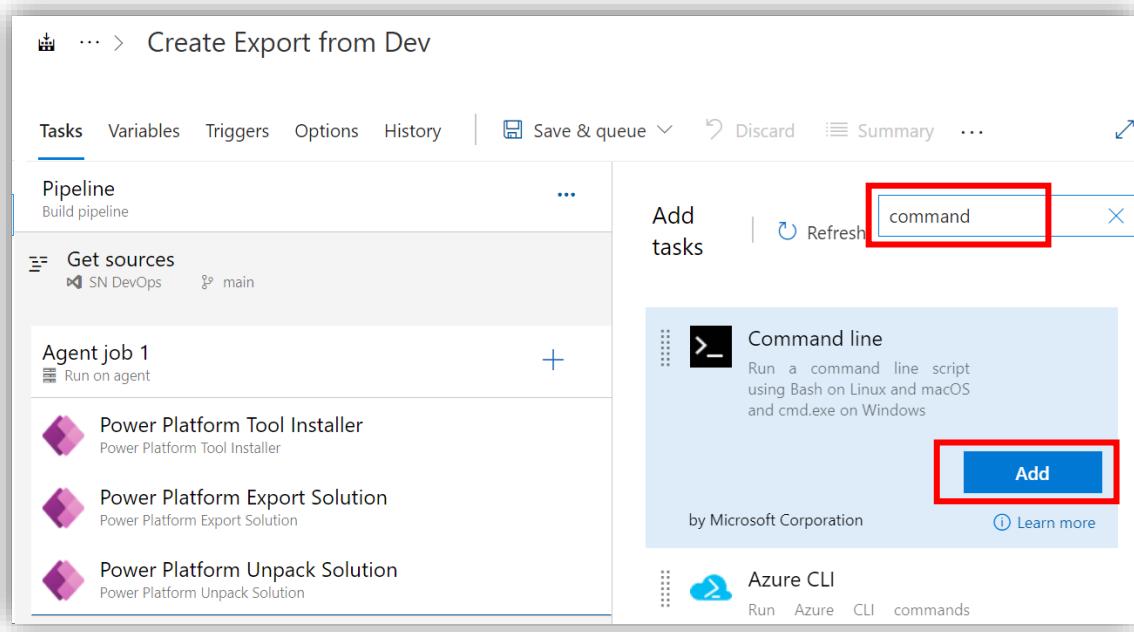


Commit Solution Files to Source Control

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.



The screenshot shows the 'Create Export from Dev' pipeline in the Azure DevOps interface. The pipeline consists of several steps:

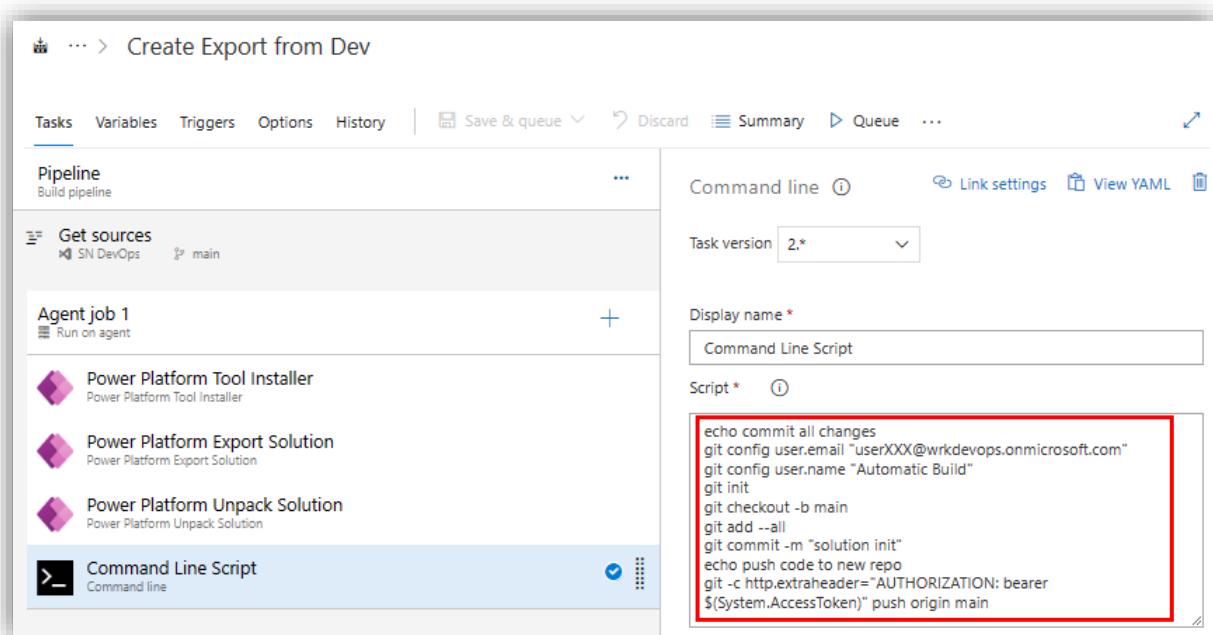
- Get sources**: SN DevOps, main
- Agent job 1**: Run on agent
 - Power Platform Tool Installer**: Power Platform Tool Installer
 - Power Platform Export Solution**: Power Platform Export Solution
 - Power Platform Unpack Solution**: Power Platform Unpack Solution

At the top right, there is a search bar with the text "command" and a red box highlighting it. Below the search bar, there is a "Add tasks" section. A card for the "Command line" task is displayed, also with a red box around its "Add" button. The "Command line" task is described as running a command line script using Bash on Linux and macOS and cmd.exe on Windows, created by Microsoft Corporation.

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

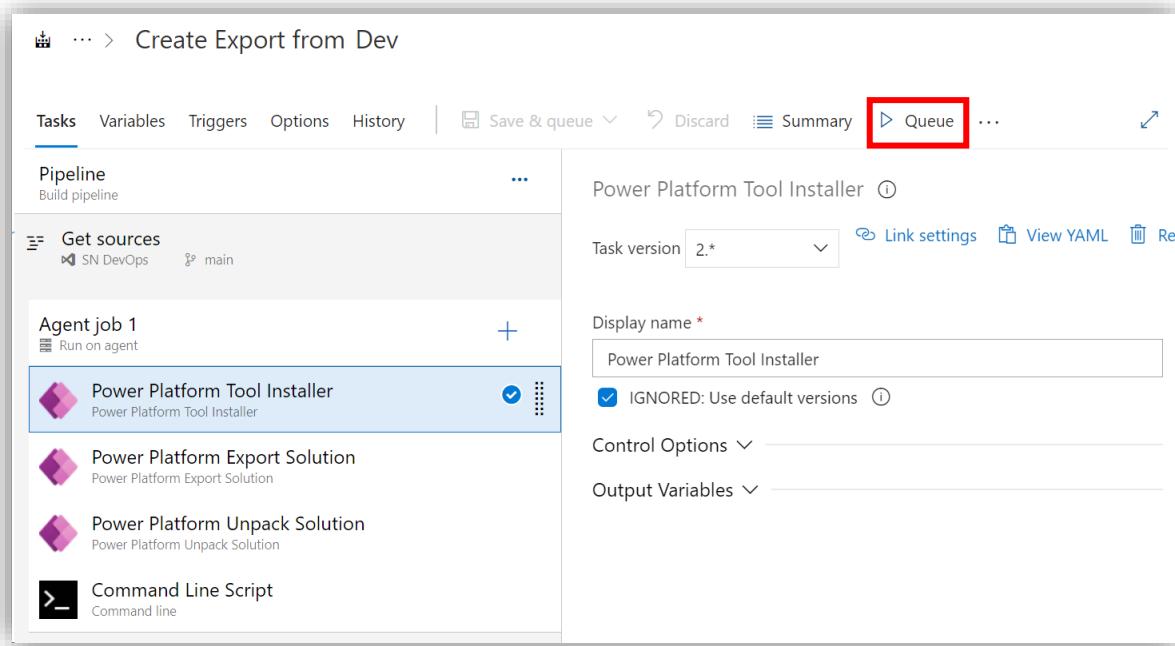
```
echo commit all changes  
git config user.email "userXXX@wrkdevops.onmicrosoft.com"  
git config user.name "Automatic Build"  
git init  
git checkout -b main  
git add --all  
git commit -m "solution init"  
echo push code to new repo  
git -c http.extraheader="AUTHORIZATION: bearer $(System.AccessToken)"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

43. Screenshot provided below:



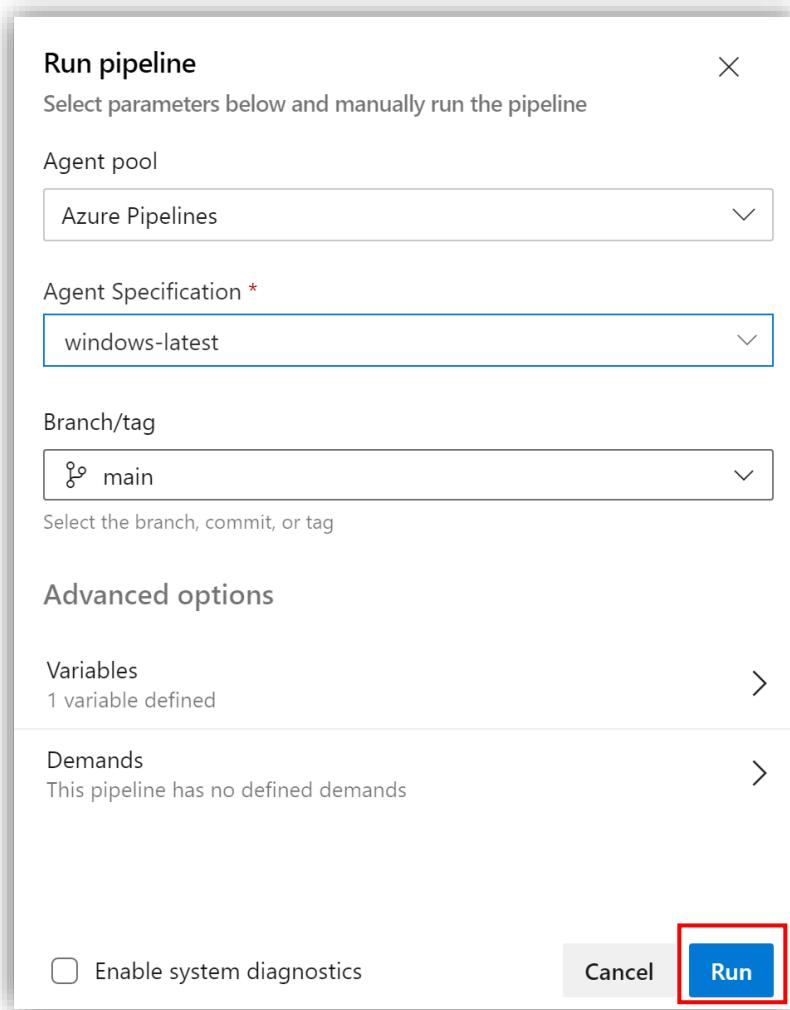
44. Save the task (**Save & queue > Save**).

Test your Pipeline

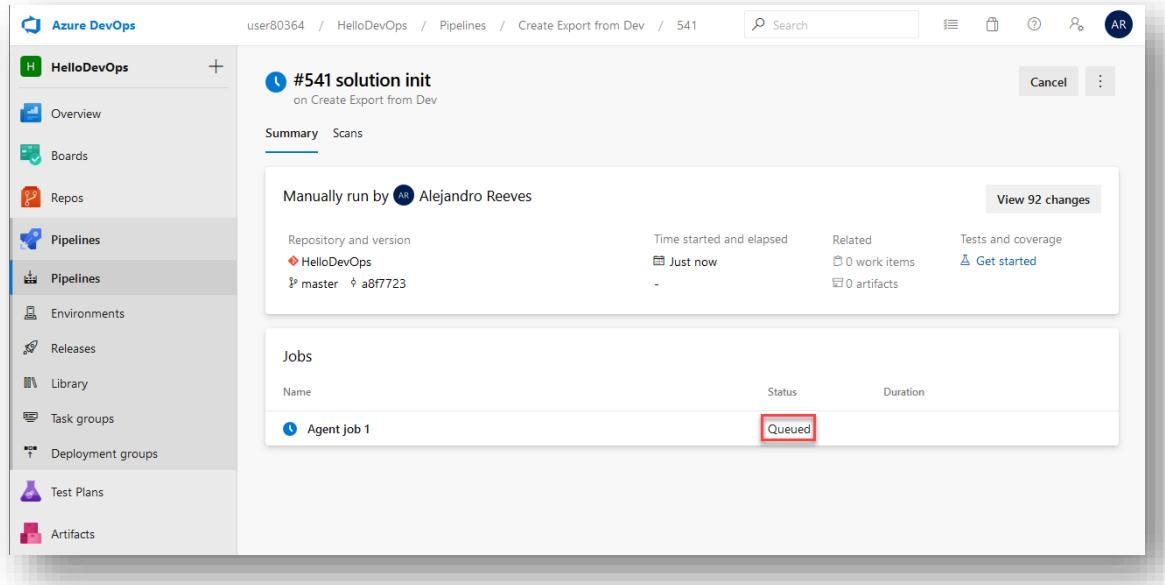
45. Select **Queue** to execute your pipeline.

The screenshot shows the 'Create Export from Dev' pipeline in the Power Automate interface. The top navigation bar includes 'Tasks', 'Variables', 'Triggers', 'Options', 'History', 'Save & queue', 'Discard', 'Summary', and a redboxed 'Queue' button. The pipeline structure on the left lists a 'Get sources' step and an 'Agent job 1' step. The 'Agent job 1' step contains four tasks: 'Power Platform Tool Installer', 'Power Platform Export Solution', 'Power Platform Unpack Solution', and 'Command Line Script'. The 'Power Platform Tool Installer' task is currently selected, indicated by a blue border. On the right side, there are sections for 'Power Platform Tool Installer' configuration, including 'Task version' set to '2.*', 'Display name' set to 'Power Platform Tool Installer', and a checked checkbox for 'IGNORED: Use default versions'. There are also sections for 'Control Options' and 'Output Variables'.

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

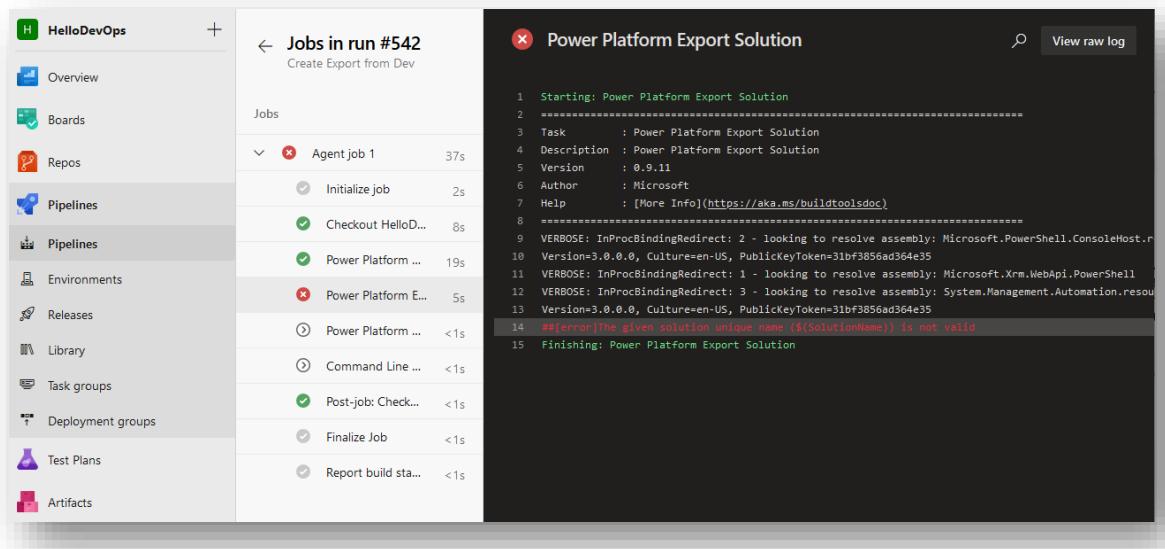


47. You can see your build has been queued. You can navigate to it directly in the notification on this screen or using the Builds area in the left navigation.



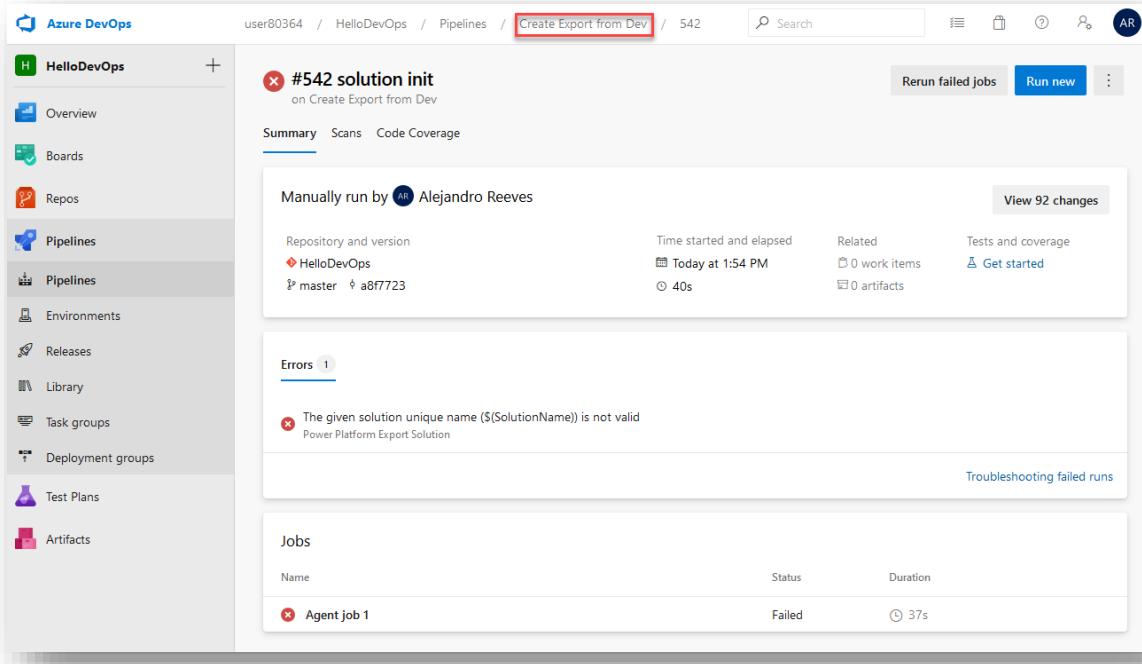
The screenshot shows the Azure DevOps interface for a pipeline named "HelloDevOps". The pipeline has a single step: "Create Export from Dev". A notification for build #541, titled "solution init", is displayed, indicating it was manually run by Alejandro Reeves and is currently in a "Queued" state. The pipeline details show it's based on the "HelloDevOps" repository, master branch, and was started just now.

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.



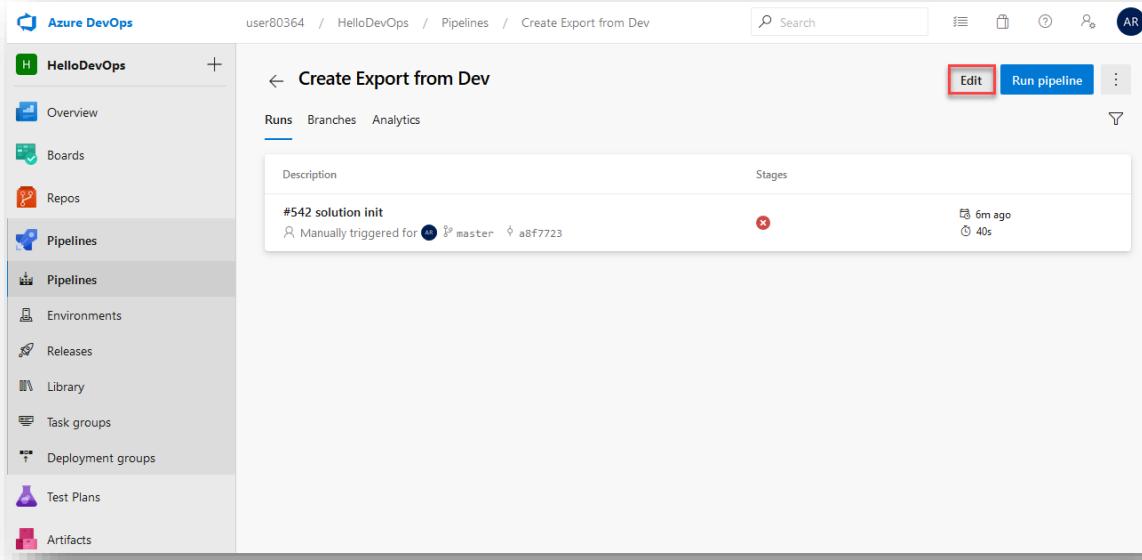
The screenshot shows the Azure DevOps interface for pipeline #542. The "Jobs" section lists several steps: "Agent job 1", "Initialize job", "Checkout HelloD...", "Power Platform ...", "Power Platform E...", "Power Platform ...", "Command Line ...", "Post-job: Check...", "Finalize Job", and "Report build sta...". The "Power Platform E..." step is marked with a red error icon. The right panel displays the log for the "Power Platform Export Solution" task, which fails at step 14 with the message "#[error]The given solution unique name \${{SolutionName}} is not valid".

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.



The screenshot shows the Azure DevOps Pipelines interface. On the left, the navigation bar includes 'HelloDevOps' (selected), Overview, Boards, Repos, Pipelines (selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area displays a failed pipeline run titled '#542 solution init' on 'Create Export from Dev'. The 'Summary' tab is selected, showing it was manually run by Alejandro Reeves. It details the repository as 'HelloDevOps' on 'master' branch at commit 'a8f7723', starting today at 1:54 PM and taking 40 seconds. There are 0 work items and 0 artifacts. An error message states: 'The given solution unique name (\$SolutionName) is not valid' for the 'Power Platform Export Solution'. The 'Jobs' section shows one job named 'Agent job 1' which failed after 37 seconds. Buttons for 'Rerun failed jobs', 'Run new', and more options are visible.

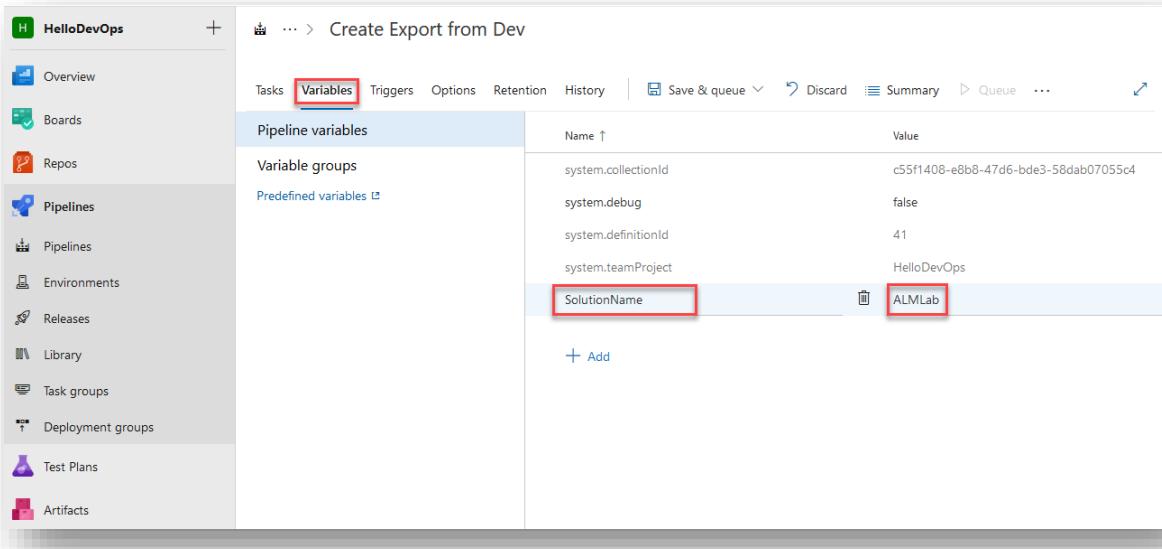
50. Click the **Edit** button.



The screenshot shows the 'Create Export from Dev' pipeline edit screen. The left sidebar has the same navigation as the previous screenshot. The main area shows the pipeline run '#542 solution init' with a status of 'Failed'. The 'Edit' button is highlighted with a red box. Other buttons include 'Run pipeline' and a three-dot menu. The pipeline run details show it was triggered manually for the 'master' branch at commit 'a8f7723' 6 minutes ago, with a duration of 40 seconds. A delete icon is next to the run entry.

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.

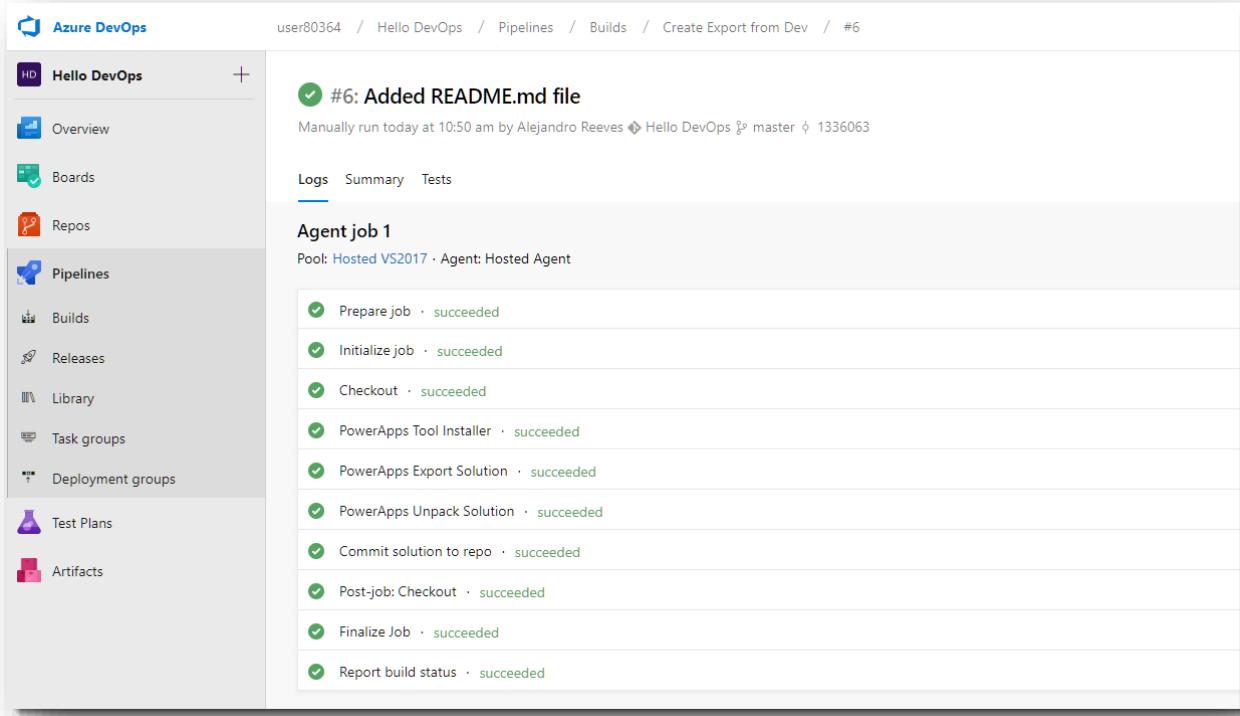


The screenshot shows the 'Variables' tab in the Azure DevOps 'Create Export from Dev' pipeline. The 'Pipeline variables' section lists several variables:

| Name ↑ | Value |
|---------------------|--------------------------------------|
| system.collectionId | c55f1408-e8b8-47d6-bde3-58dab07055c4 |
| system.debug | false |
| system.definitionId | 41 |
| system.teamProject | HelloDevOps |
| SolutionName | ALMLab |

A red box highlights the 'SolutionName' row, and another red box highlights the value 'ALMLab'.

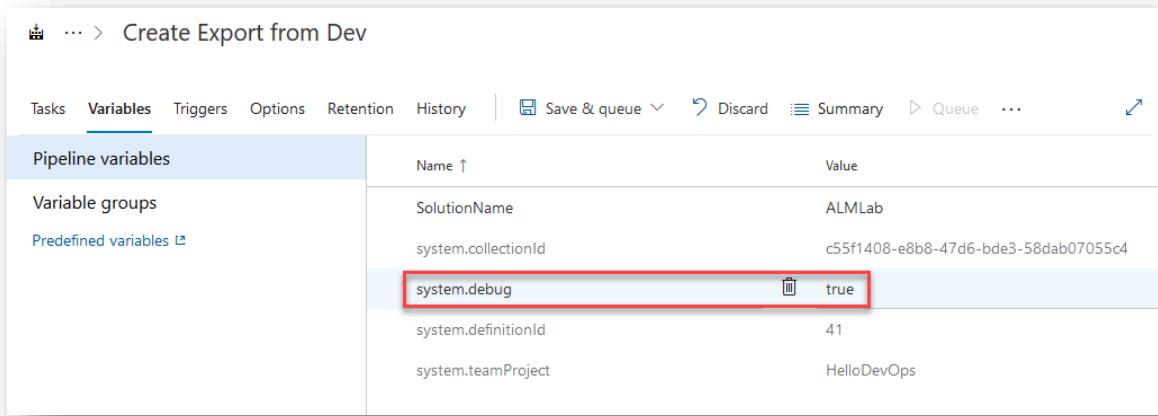
53. Click **Save & queue** and observe the results. Fix any issues until you get a successful build.



The screenshot shows the Azure DevOps interface for a pipeline named "Hello DevOps". The pipeline has completed successfully, indicated by a green checkmark icon and the text "#6: Added README.md file". The build was manually run today at 10:50 am by Alejandro Reeves. The pipeline details show the following steps:

- Agent job 1 (Hosted VS2017 - Hosted Agent)
 - Prepare job · succeeded
 - Initialize job · succeeded
 - Checkout · succeeded
 - PowerApps Tool Installer · succeeded
 - PowerApps Export Solution · succeeded
 - PowerApps Unpack Solution · succeeded
 - Commit solution to repo · succeeded
 - Post-job: Checkout · succeeded
 - Finalize Job · succeeded
 - Report build status · succeeded

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

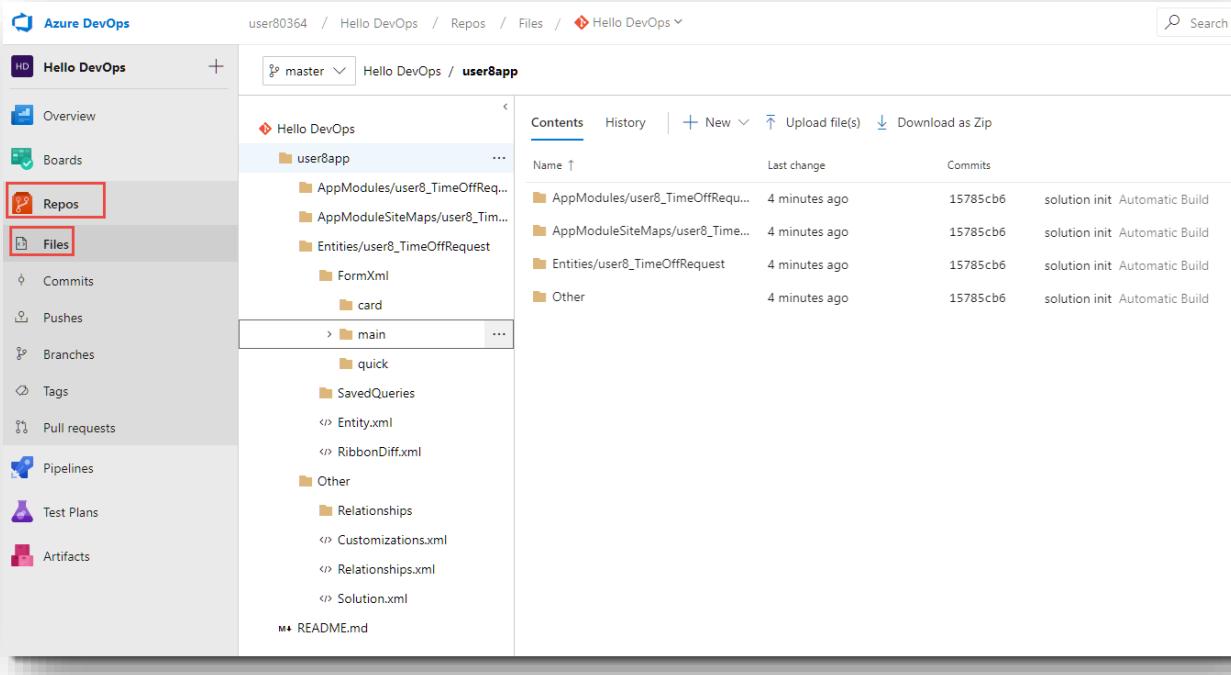


The screenshot shows the "Variables" tab for the "Create Export from Dev" pipeline. The "Pipeline variables" section lists several variables:

| Name ↑ | Value |
|---------------------|--------------------------------------|
| SolutionName | ALMLab |
| system.collectionId | c55f1408-e8b8-47d6-bde3-58dab07055c4 |
| system.debug | true |
| system.definitionId | 41 |
| system.teamProject | HelloDevOps |

The "system.debug" row is highlighted with a red box around the "true" value.

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.



The screenshot shows the Azure DevOps interface for a repository named 'Hello DevOps'. The left sidebar has a 'Repos' button highlighted with a red box. The main area shows a file tree for the 'user8app' folder. The 'Contents' tab is selected. The file tree includes:

- AppModules/user8_TimeOffReq...
- AppModuleSiteMaps/user8_Tim...
- Entities/user8_TimeOffRequest
 - FormXml
 - card
 - main
 - quick
 - SavedQueries
 - Entity.xml
 - RibbonDiff.xml
 - Other
 - Relationships
 - Customizations.xml
 - Relationships.xml
 - Solution.xml
- README.md

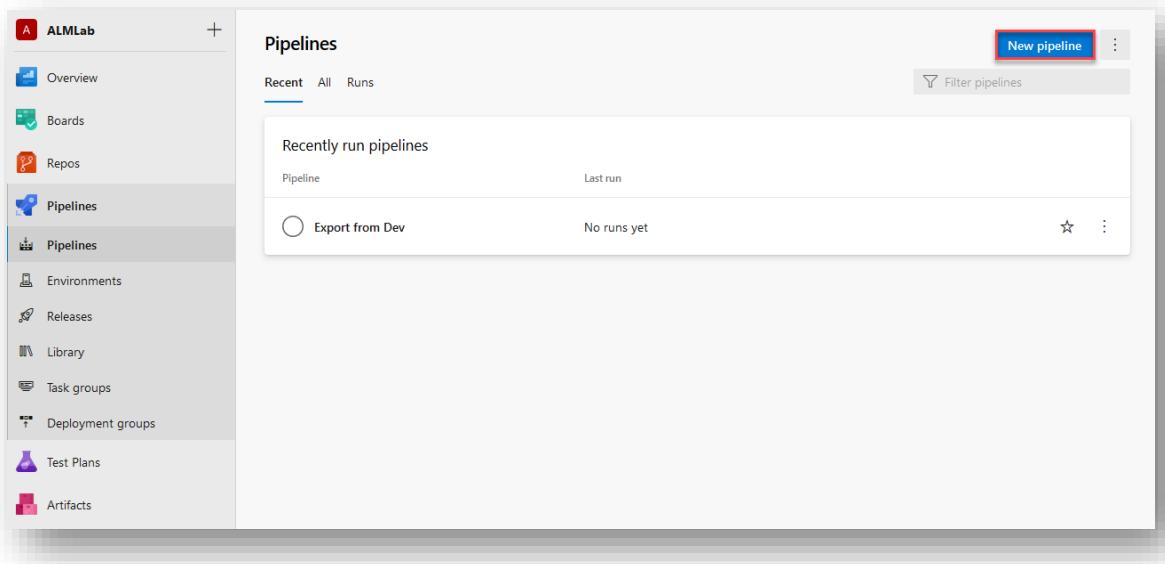
The right side of the interface shows a table of commits for the 'master' branch:

| Name | Last change | Commits |
|--------------------------------|---------------|--|
| AppModules/user8_TimeOffReq... | 4 minutes ago | 15785cb6 solution init Automatic Build |
| AppModuleSiteMaps/user8_Tim... | 4 minutes ago | 15785cb6 solution init Automatic Build |
| Entities/user8_TimeOffRequest | 4 minutes ago | 15785cb6 solution init Automatic Build |
| Other | 4 minutes ago | 15785cb6 solution init Automatic Build |

Pipeline 2: Build your Managed Solution

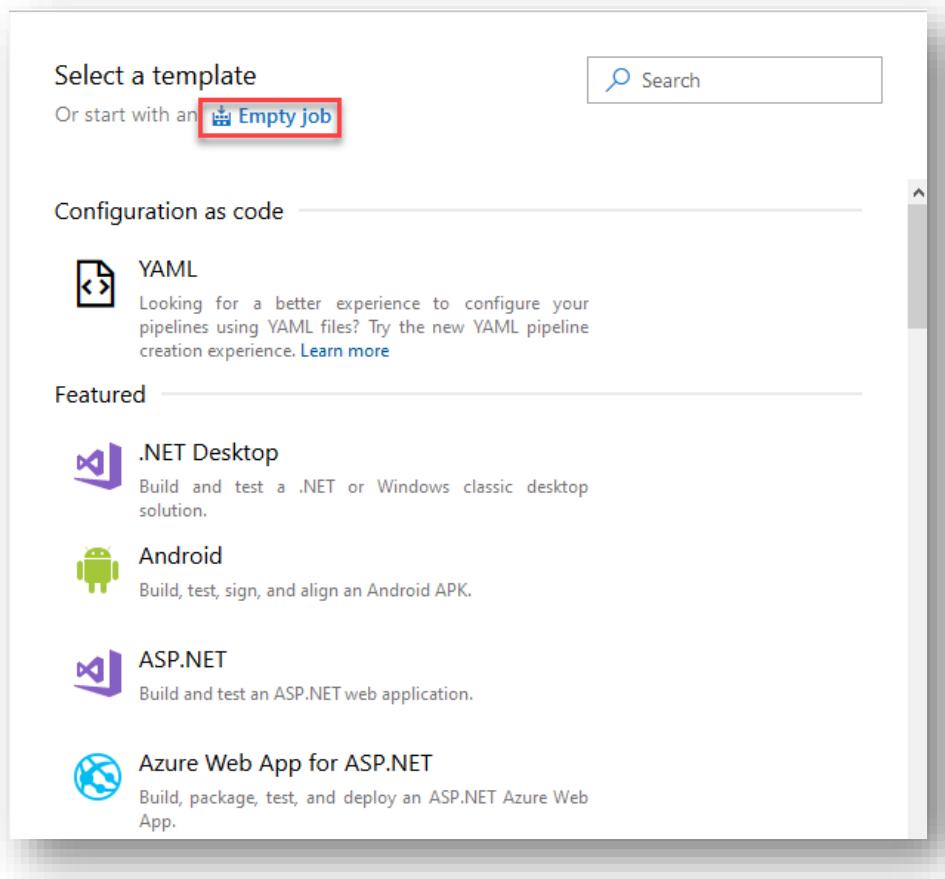
Your unmanaged solution is checked into source control, but you need to have a managed solution to deploy to other environments such as test and production. To obtain your managed solution, you should use a Just-In-Time (JIT) build environment where you would import your unmanaged solution files then 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.

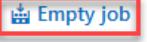


The screenshot shows the PowerPlatform ALM Workshop interface. On the left, there is a sidebar with various navigation options: Overview, Boards, Repos, Pipelines (which is selected and highlighted in blue), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area is titled "Pipelines" and contains a section for "Recently run pipelines". It shows one pipeline named "Export from Dev" with the status "No runs yet". At the top right of the main area, there is a "New pipeline" button, which is highlighted with a red box. Below it is a "Filter pipelines" search bar.

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 job** link on the template selection page shown below.



Select a template

Or start with an [Empty job](#) 

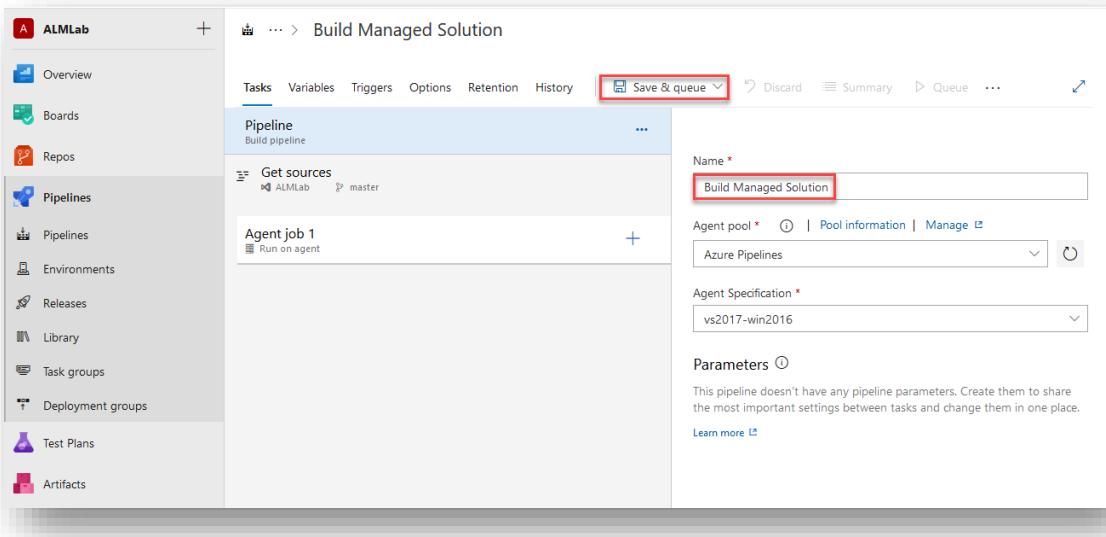
Configuration as code [YAML](#)

Looking for a better experience to configure your pipelines using YAML files? Try the new YAML pipeline creation experience. [Learn more](#)

Featured

-  .NET Desktop
Build and test a .NET or Windows classic desktop solution.
-  Android
Build, test, sign, and align an Android APK.
-  ASP.NET
Build and test an ASP.NET web application.
-  Azure Web App for ASP.NET
Build, package, test, and deploy an ASP.NET Azure Web App.

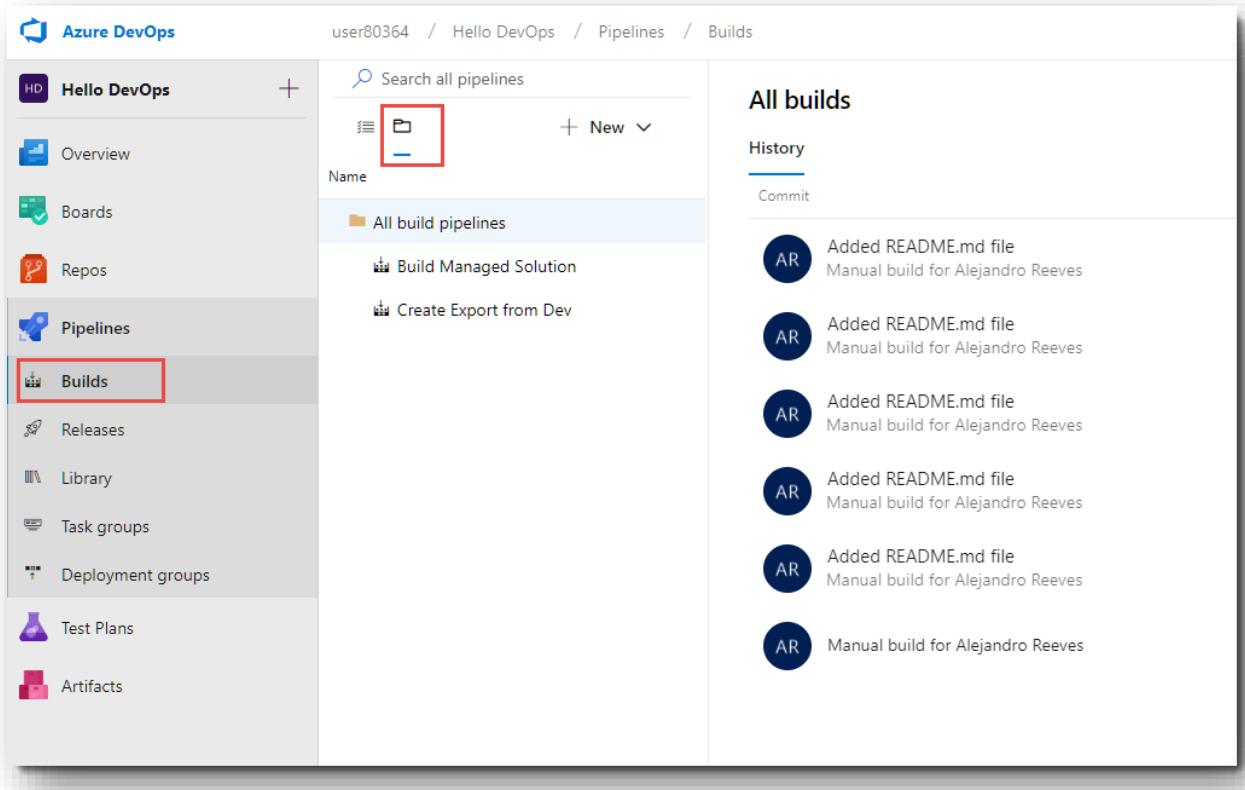
57. Name the pipeline “Build Managed Solution” and save it. This will give you an empty pipeline.



The screenshot shows the Azure Pipelines interface for creating a new build pipeline. The left sidebar is titled 'ALMLab' and includes options like Overview, Boards, Repos, Pipelines (which is selected), Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area is titled 'Build Managed Solution' and shows a single step: 'Agent job 1'. The 'Save & queue' button is highlighted with a red box. The pipeline configuration includes:

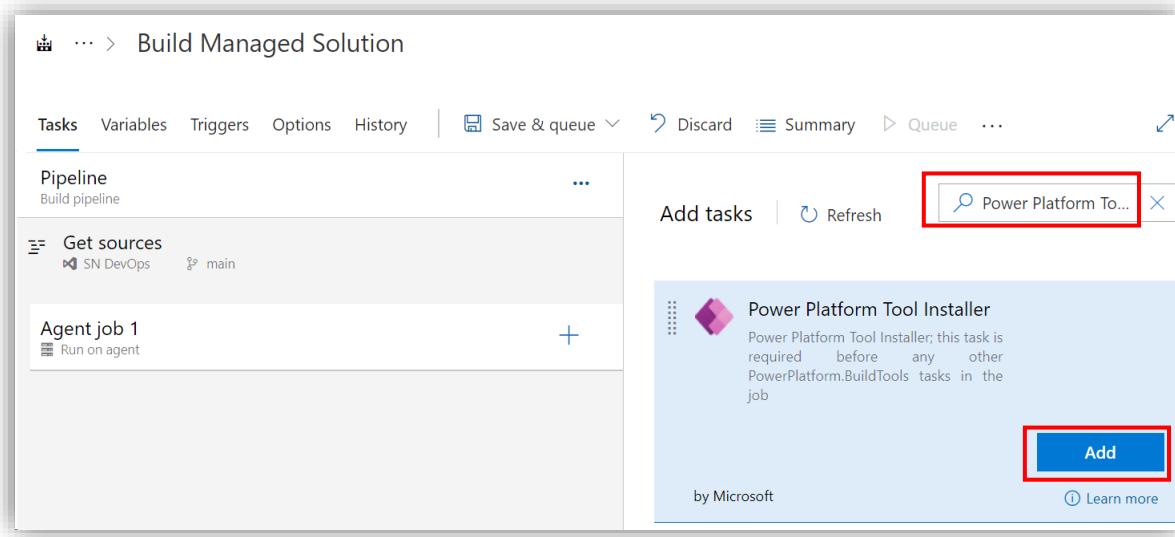
- Name:** Build Managed Solution
- Agent pool:** Azure Pipelines
- Agent Specification:** vs2017-win2016
- Parameters:** None (This pipeline doesn't have any pipeline parameters. Create them to share the most important settings between tasks and change them in one place.)

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.



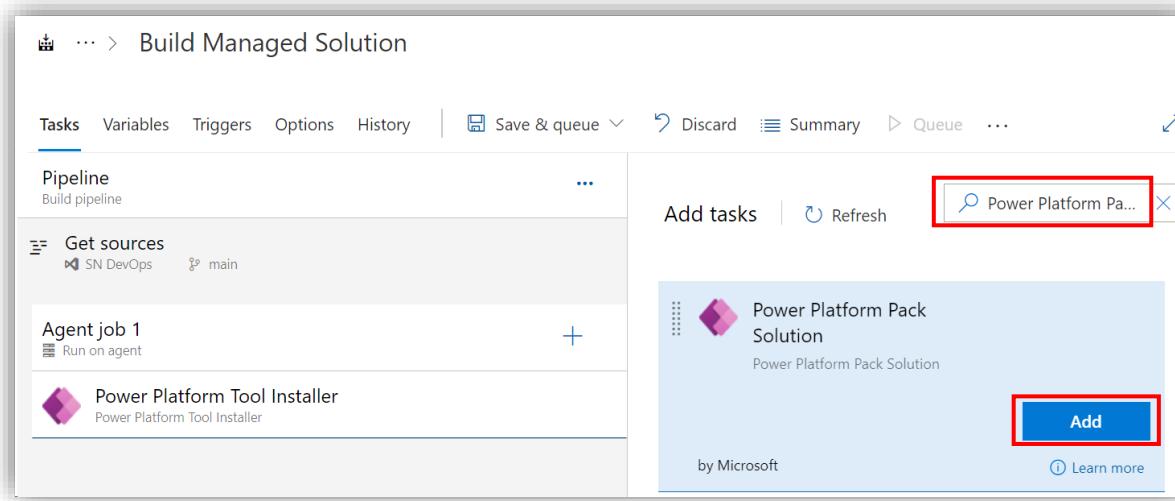
The screenshot shows the Azure DevOps interface for the 'Hello DevOps' project. The left sidebar is open, showing options like Overview, Boards, Repos, Pipelines, Builds (which is selected and highlighted with a red box), Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main content area is titled 'All builds' and shows a history of manual builds for Alejandro Reeves (AR). Each build entry includes a blue circular profile picture with 'AR', the build name ('Added README.md file'), and a description ('Manual build for Alejandro Reeves'). Above the build list, there's a search bar labeled 'Search all pipelines' and a 'New' button. The top navigation bar shows the user 'user80364' and the project 'Hello DevOps'.

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



The screenshot shows the Azure DevOps Pipeline interface for a "Build Managed Solution" pipeline. On the left, there's a sidebar with "Tasks", "Variables", "Triggers", "Options", and "History". The main area shows a "Pipeline" section with a "Get sources" step and an "Agent job 1" step. In the "Agent job 1" step, there's a plus sign to add more tasks. On the right, a modal window titled "Add tasks" is open, showing a search bar with "Power Platform To..." and a result for "Power Platform Tool Installer". This result has a blue "Add" button highlighted with a red box.

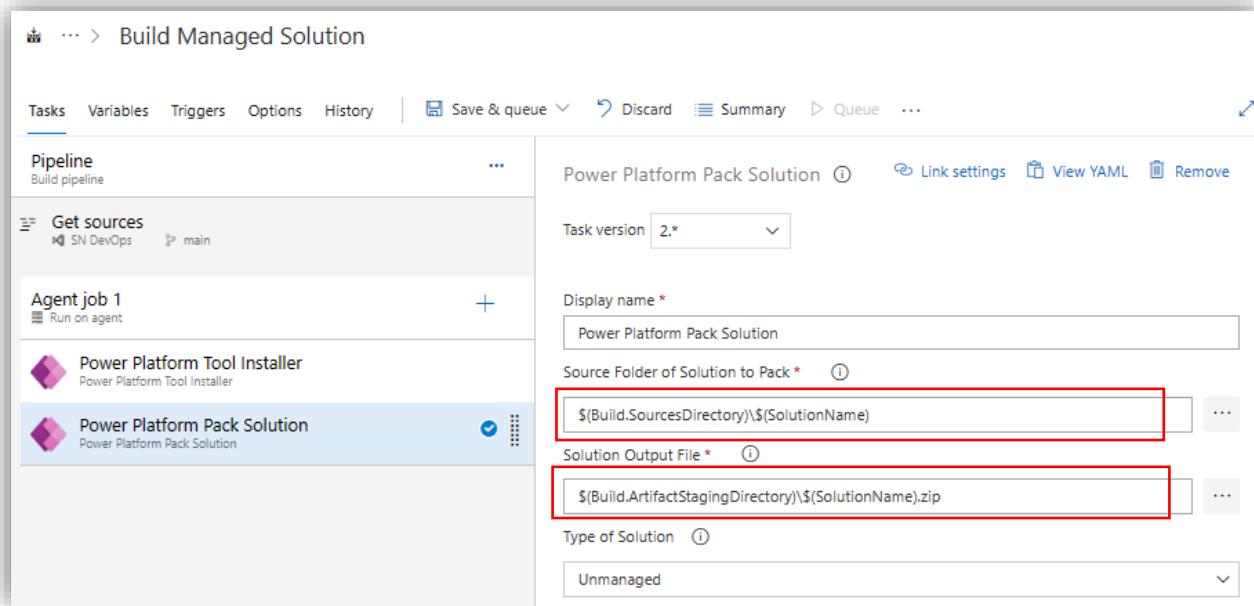
60. Add the **Power Platform Pack Solution** task.



The screenshot shows the Azure DevOps Pipeline interface for a "Build Managed Solution" pipeline. The layout is identical to the previous screenshot, with a sidebar and a main pipeline area. The "Agent job 1" step now includes the previously added "Power Platform Tool Installer" task. A new "Power Platform Pack Solution" task is listed below it. The search bar in the "Add tasks" modal is now showing "Power Platform Pa...", and the "Power Platform Pack Solution" result also has a blue "Add" button highlighted with a red box.

61. Configure the pack solution task with the following parameters:

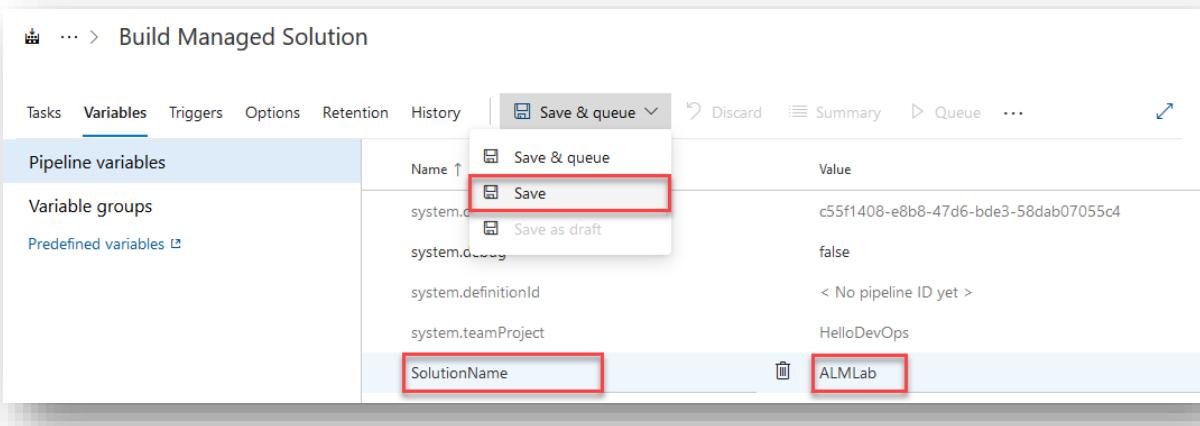
- Source Folder of Solution to Pack: `$(Build.SourcesDirectory)\$(SolutionName)`
- Solution Output File: `$(Build.ArtifactStagingDirectory)\$(SolutionName).zip`



The screenshot shows the 'Build Managed Solution' pipeline in Azure DevOps. The pipeline consists of three tasks: 'Get sources', 'Agent job 1' (containing 'Power Platform Tool Installer'), and 'Power Platform Pack Solution'. The 'Power Platform Pack Solution' task is selected. Its configuration includes:

- Task version: 2.*
- Display name: Power Platform Pack Solution
- Source Folder of Solution to Pack: `$(Build.SourcesDirectory)\$(SolutionName)`
- Solution Output File: `$(Build.ArtifactStagingDirectory)\$(SolutionName).zip`
- Type of Solution: Unmanaged

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).

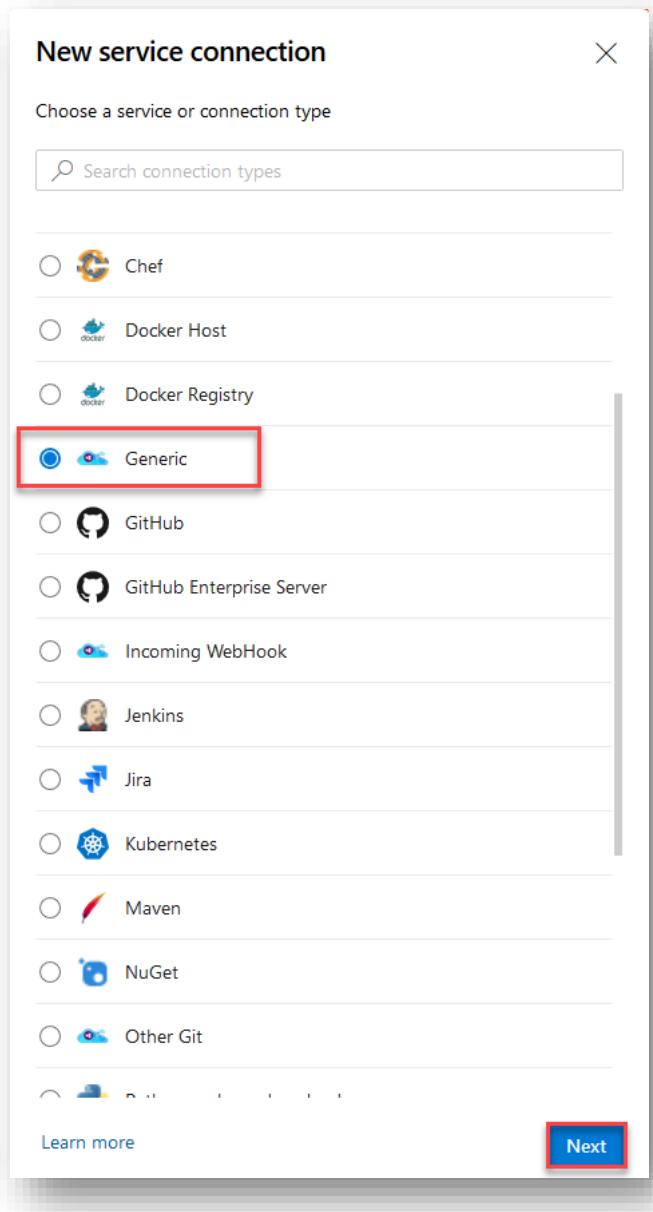


| Name | Value |
|---------------------|------------------------|
| system.debug | false |
| system.definitionId | < No pipeline ID yet > |
| system.teamProject | HelloDevOps |
| SolutionName | ALMLab |

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**.

The screenshot shows the 'Project Settings' page for the 'ALMLab' project in the ALM Lab interface. The left sidebar lists various settings sections: Overview, Boards, Repos, Pipelines, Test Plans, and Artifacts. The 'Project settings' tab is selected and highlighted with a red box. The main content area is titled 'Service connections'. At the top right of this area, there is a button labeled 'New service connection' which is also highlighted with a red box. Below this button is a search bar with the placeholder 'Filter by keywords' and a dropdown menu set to 'Created by'. A single service connection named 'Development' is listed. The bottom of the page shows a navigation bar with back and forward arrows.

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.

New Generic service connection

Server URL
https://ppbuildtoolsbuild.crm.dynamics.com

Authentication

Username (optional) Password/Token Key (optional)
admin@almlab.onmicrosoft.com ··········
Username for connecting to the endpoint Password/Token Key for connecting to the endpoint

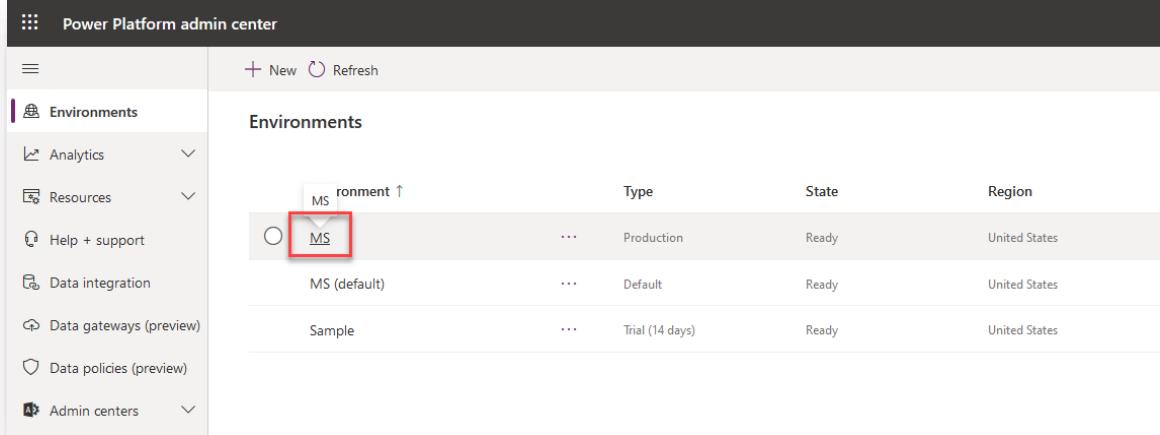
Details

Service connection name
Build

Description (optional)
Lab build environment

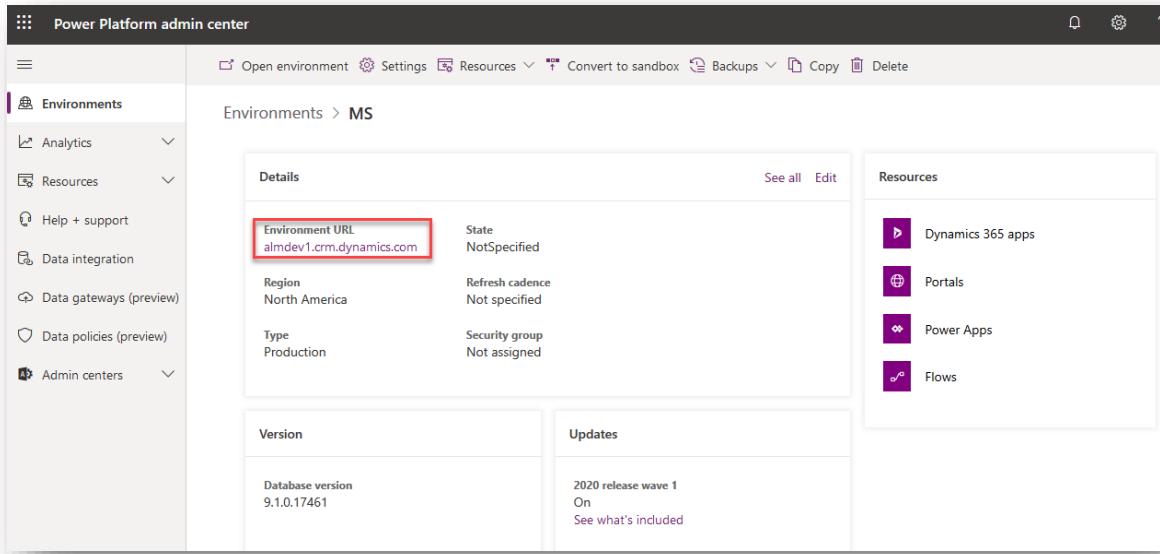
Security
 Grant access permission to all pipelines

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.



The screenshot shows the 'Environments' section of the Power Platform admin center. A red box highlights the 'MS' environment in the list, which is currently selected. The table displays three environments: MS (selected), MS (default), and Sample.

| Environment | Type | State | Region |
|--------------|-----------------|-------|---------------|
| MS | Production | Ready | United States |
| MS (default) | Default | Ready | United States |
| Sample | Trial (14 days) | Ready | United States |



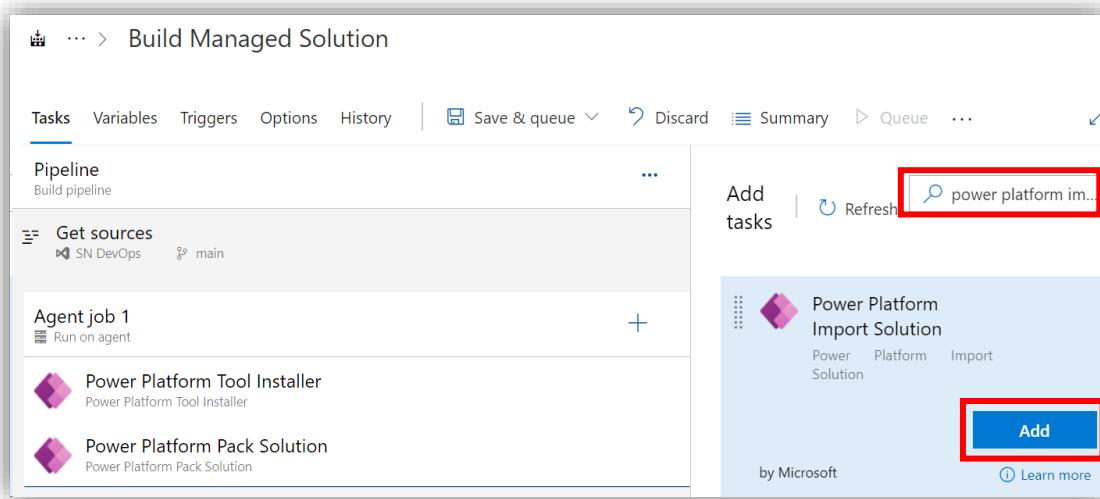
The screenshot shows the details for the 'MS' environment. The 'Environment URL' field is highlighted with a red box and contains the value 'almdev1.crm.dynamics.com'. The 'Details' section also shows Region: North America, Refresh cadence: Not specified, Type: Production, and Security group: Not assigned. The 'Resources' sidebar on the right lists Dynamics 365 apps, Portals, Power Apps, and Flows.

| Details | See all | Edit |
|---|----------------------------------|------|
| Environment URL almdev1.crm.dynamics.com | State NotSpecified | |
| Region North America | Refresh cadence Not specified | |
| Type Production | Security group Not assigned | |

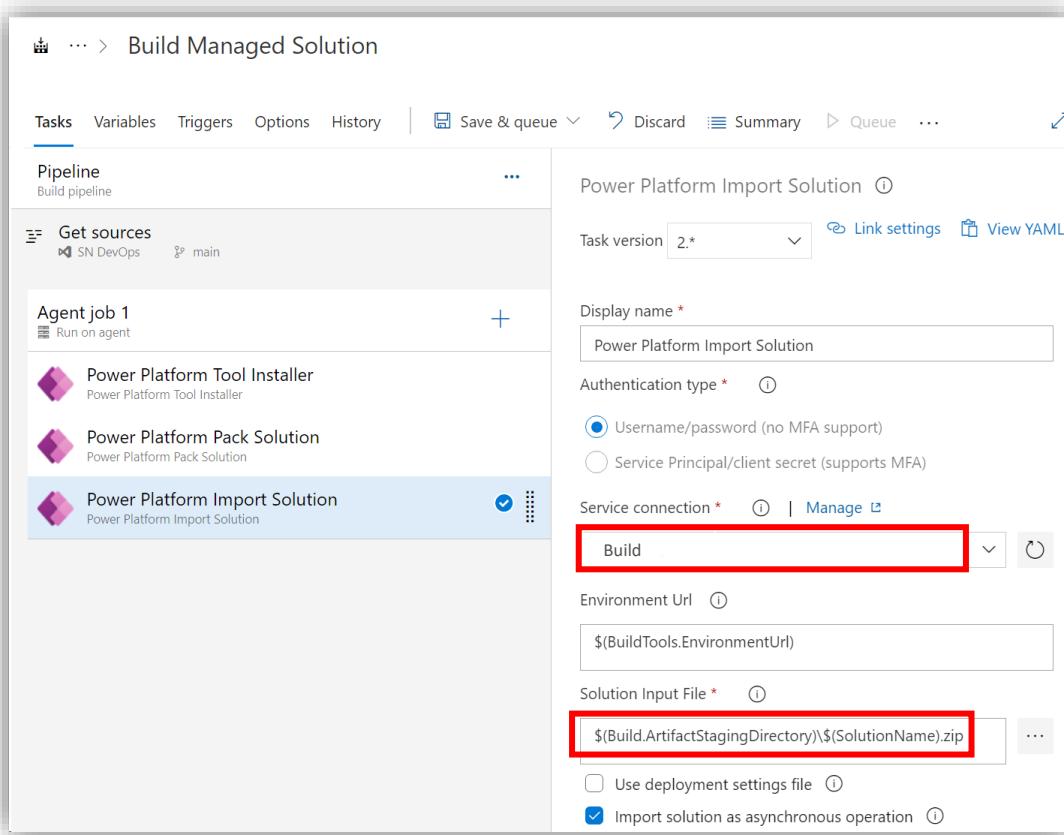
| Version | Updates |
|---------------------------------|--|
| Database version 9.1.0.17461 | 2020 release wave 1 On See what's included |

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

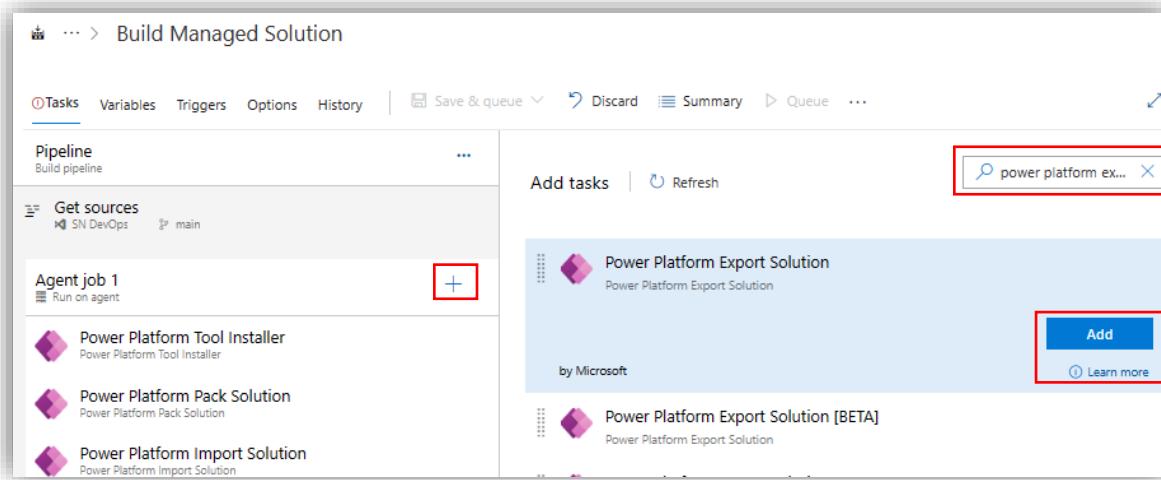
67. Navigate back to your Build Managed Solution pipeline (If you don't see the pipeline go to step 57 on how to view all pipelines). Click **Edit**, and then add the **Power Platform Import Solution** task to take the solution file and import it into your build environment.



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



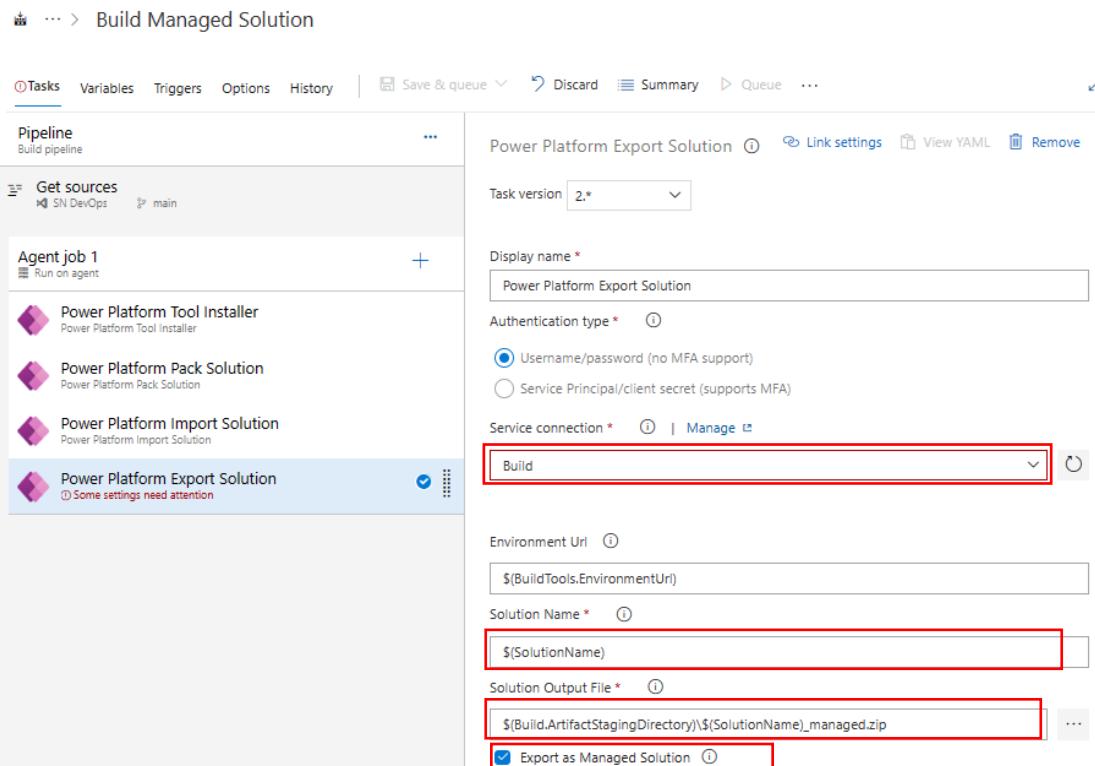
69. Add a task to export the solution file as a managed file.



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**.

Build Managed Solution

Tasks Variables Triggers Options History Save & queue Discard Summary Queue ...

Pipeline
Build pipeline

Get sources
SN DevOps main

Agent job 1 Run on agent

+
Power Platform Tool Installer
Power Platform Pack Solution
Power Platform Import Solution
Power Platform Export Solution Some settings need attention

Power Platform Export Solution

Task version 2.*

Display name * Power Platform Export Solution

Authentication type * Username/password (no MFA support) Service Principal/client secret (supports MFA)

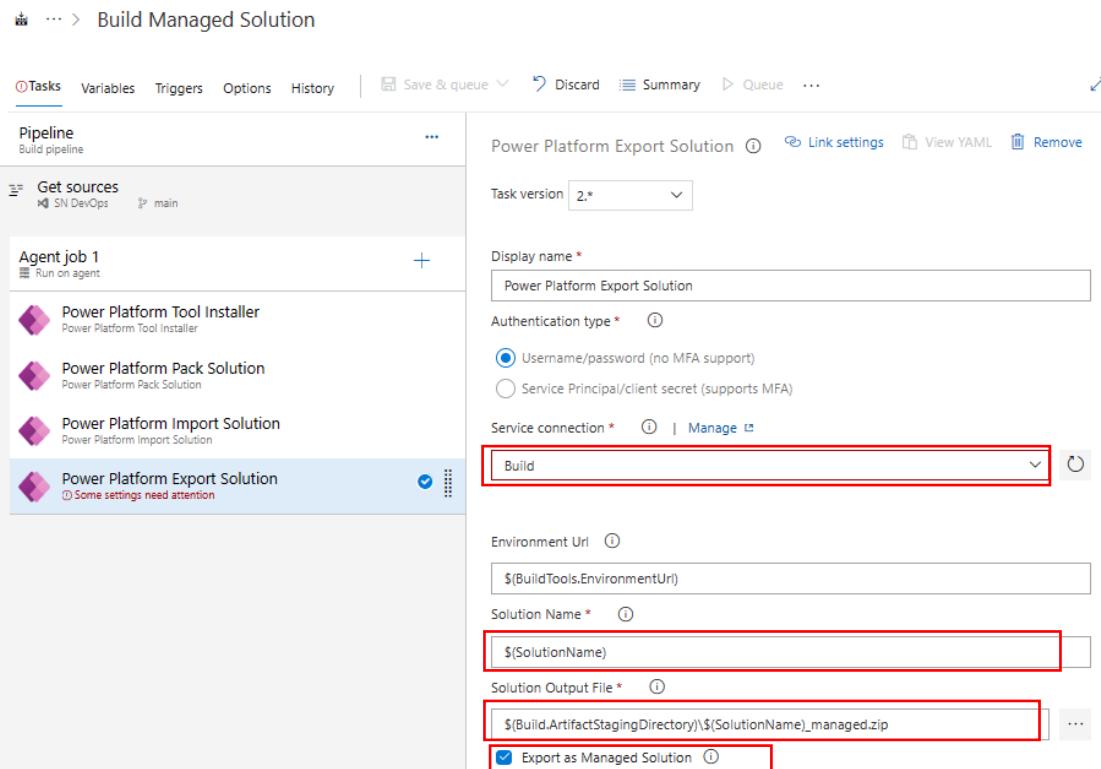
Service connection * Manage Build

Environment Url \$(BuildTools.EnvironmentUrl)

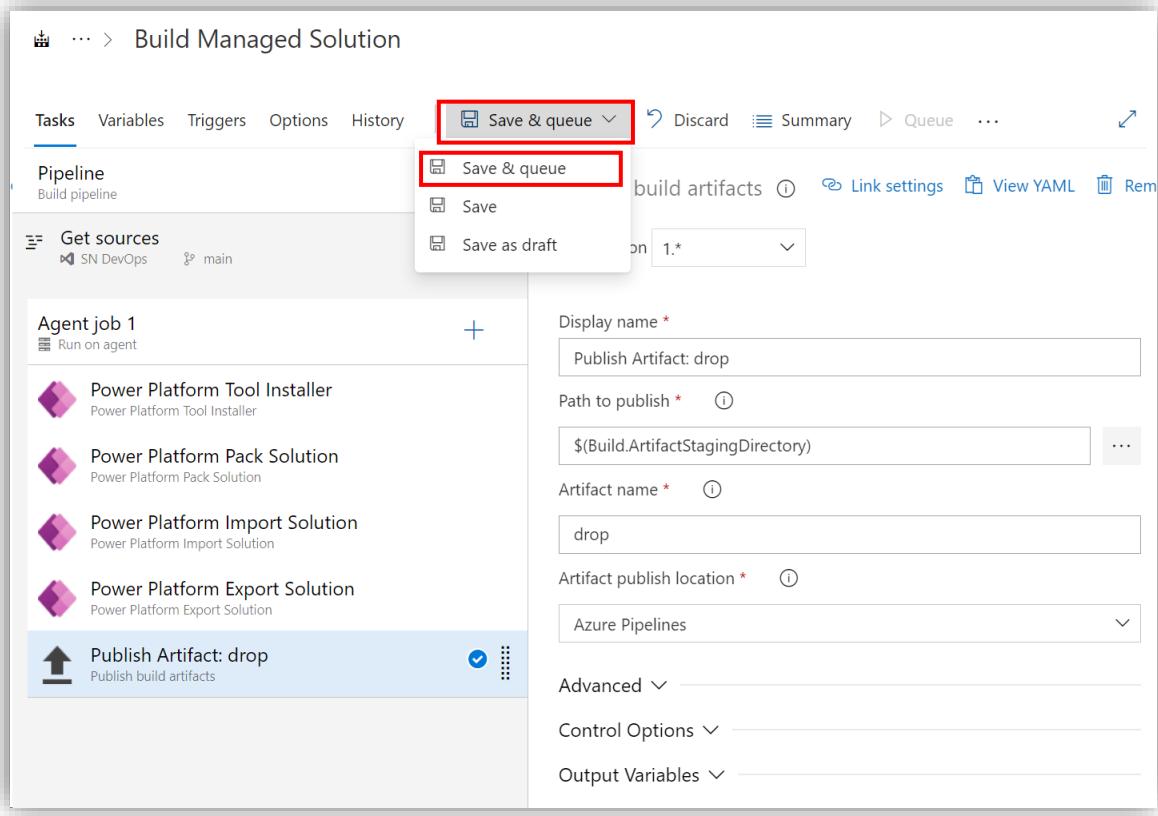
Solution Name * \$(SolutionName)

Solution Output File * \$(Build.ArtifactStagingDirectory)\\$(SolutionName)_managed.zip

Export as Managed Solution

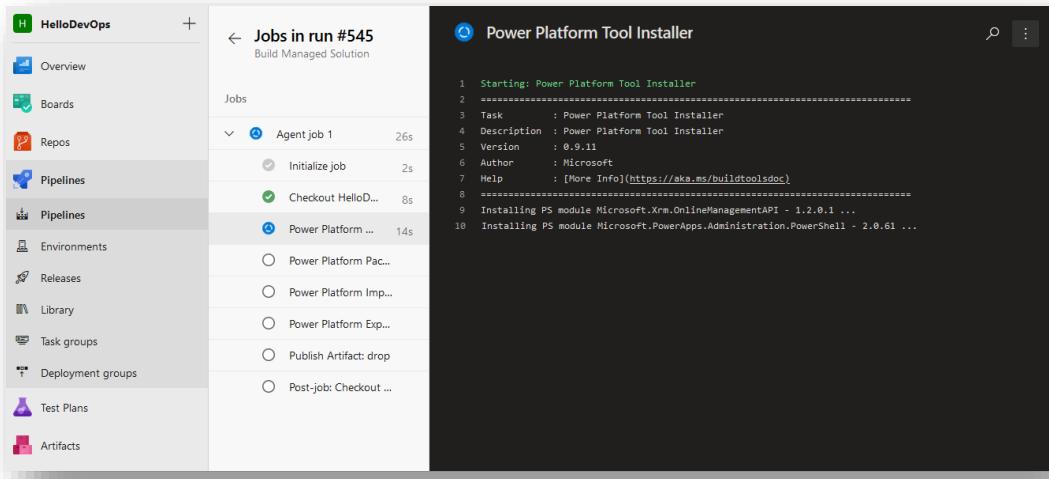


72. Leave the predefined values as-is and click **Save & queue**:



The screenshot shows the Azure DevOps interface for a 'Build Managed Solution' pipeline. The top navigation bar includes 'Tasks', 'Variables', 'Triggers', 'Options', 'History', and a 'Save & queue' dropdown. The 'Save & queue' button is highlighted with a red box. A dropdown menu from this button also has 'Save & queue' highlighted. Other options in the dropdown are 'Save' and 'Save as draft'. The pipeline configuration shows a 'Get sources' step (SN DevOps) and an 'Agent job 1' step. The 'Agent job 1' step contains four tasks: 'Power Platform Tool Installer', 'Power Platform Pack Solution', 'Power Platform Import Solution', and 'Power Platform Export Solution'. Below these is a 'Publish Artifact: drop' step. On the right side of the screen, there is a detailed configuration panel for the 'Publish Artifact: drop' step. It includes fields for 'Display name *' (set to 'Publish Artifact: drop'), 'Path to publish *' (set to '\$(Build.ArtifactStagingDirectory)'), 'Artifact name *' (set to 'drop'), and 'Artifact publish location *' (set to 'Azure Pipelines'). There are also sections for 'Advanced', 'Control Options', and 'Output Variables'.

73. Monitor it until it is successful.



The screenshot shows the Azure DevOps interface for monitoring a pipeline run. The left sidebar lists various project management and development tools like Boards, Repos, Pipelines, Environments, etc. The main area shows the 'Jobs in run #545' for a 'Build Managed Solution' pipeline. One job, 'Agent job 1', is expanded to show its individual tasks. One of these tasks, 'Power Platform Tool Installer', is selected and its log output is displayed in a large window on the right. The log output shows the following text:

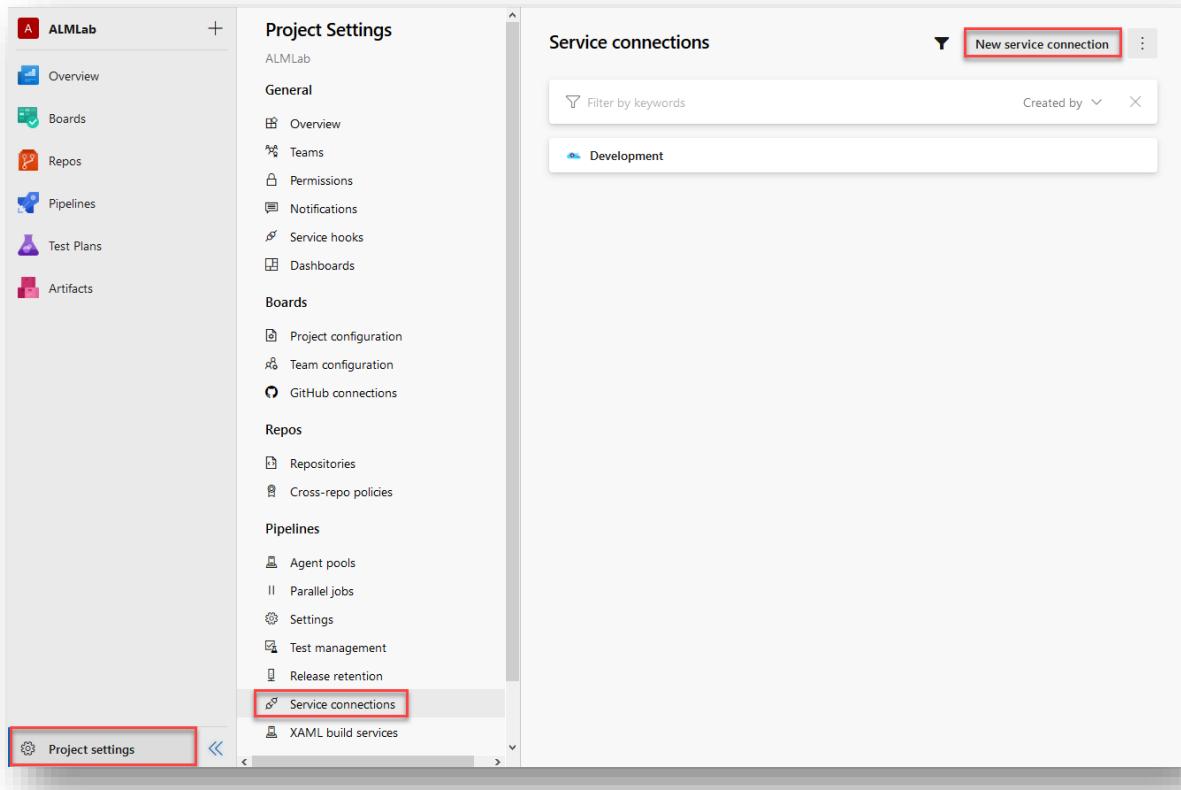
```

1 Starting: Power Platform Tool Installer
2 ****
3 Task : Power Platform Tool Installer
4 Description : Power Platform Tool Installer
5 Version : 0.9.11
6 Author : Microsoft
7 Help : [More Info](https://aka.ms/buildtoolsdoc)
8 ****
9 Installing PS module Microsoft.Xrm.OnlineManagementAPI - 1.2.0.1 ...
10 Installing PS module Microsoft.PowerApps.Administration.PowerShell - 2.0.61 ...

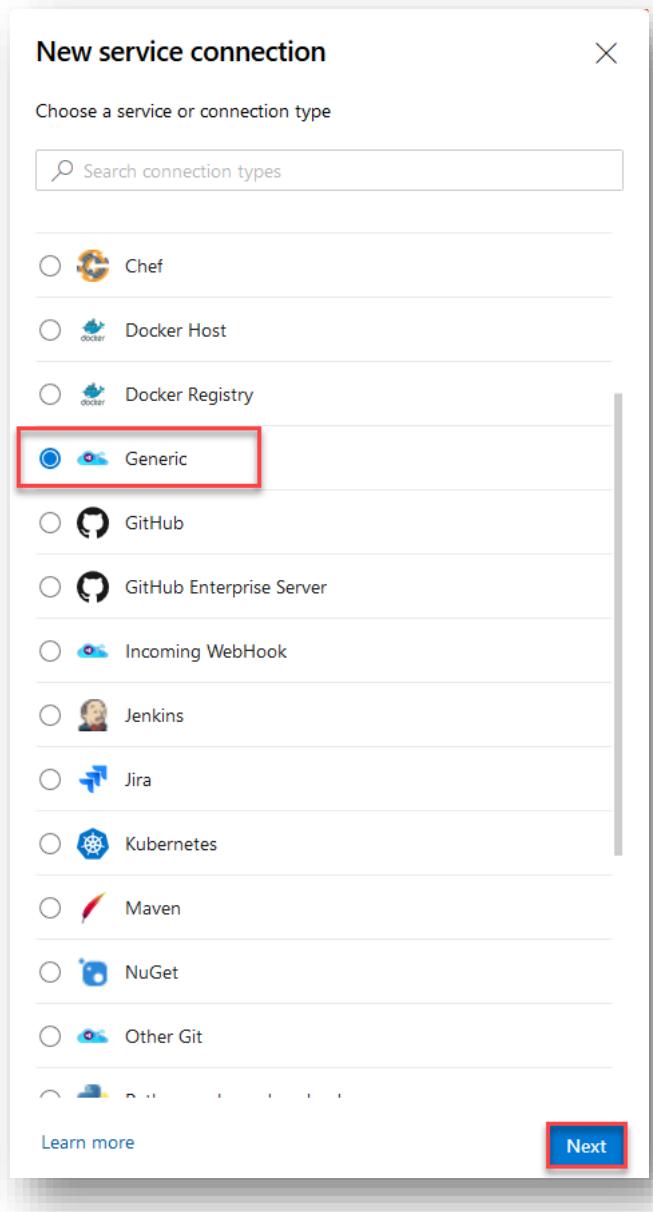
```

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.

New Generic service connection X

Server URL

Authentication

Username (optional)
Username for connecting to the endpoint

Password/Token Key (optional)
Password/Token Key for connecting to the endpoint

Details

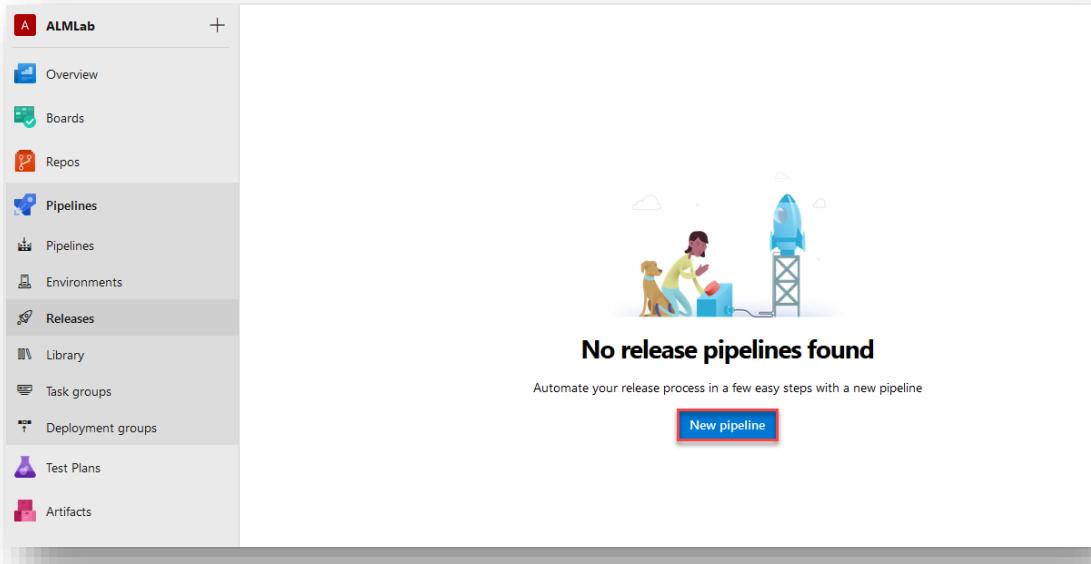
Service connection name

Description (optional)

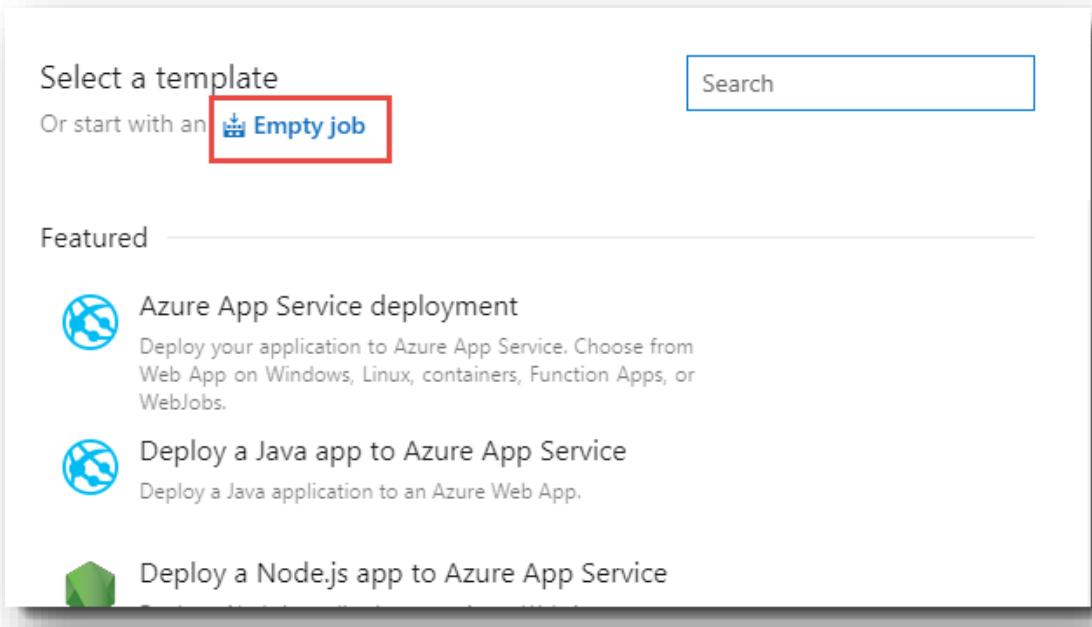
Security
 Grant access permission to all pipelines

[Learn more](#) [Troubleshoot](#) [Back](#) [Save](#)

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.



Select a template

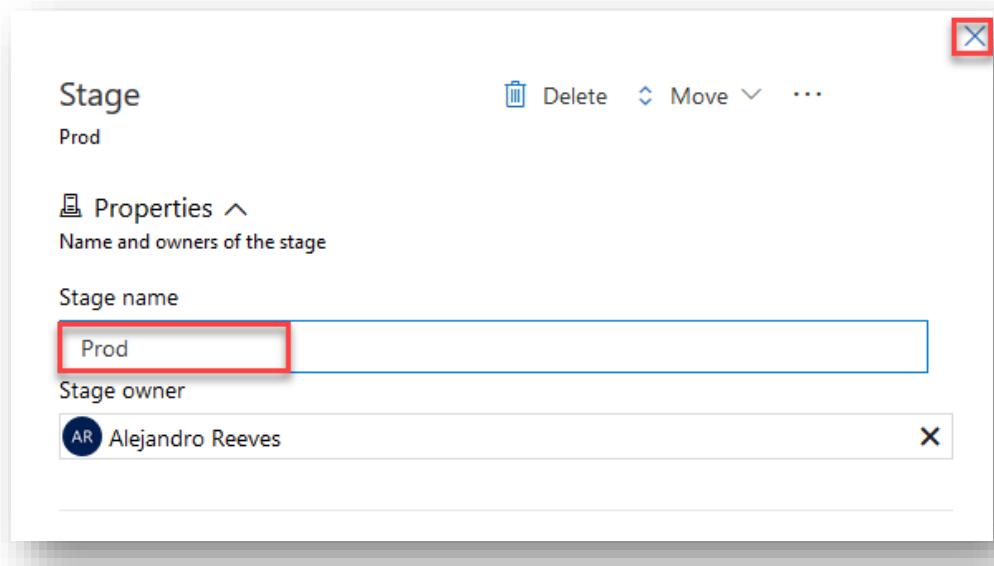
Or start with an [Empty job](#)

Search

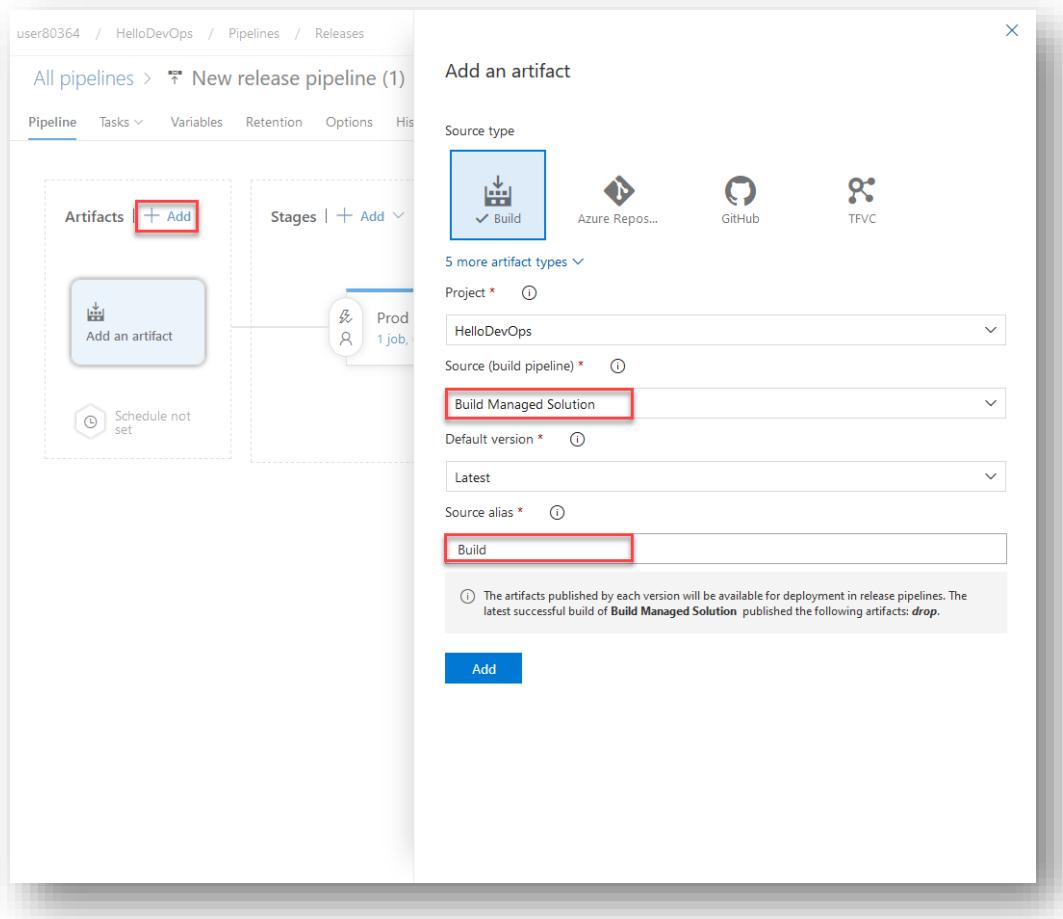
Featured

-  [Azure App Service deployment](#)
Deploy your application to Azure App Service. Choose from Web App on Windows, Linux, containers, Function Apps, or WebJobs.
-  [Deploy a Java app to Azure App Service](#)
Deploy a Java application to an Azure Web App.
-  [Deploy a Node.js app to Azure App Service](#)

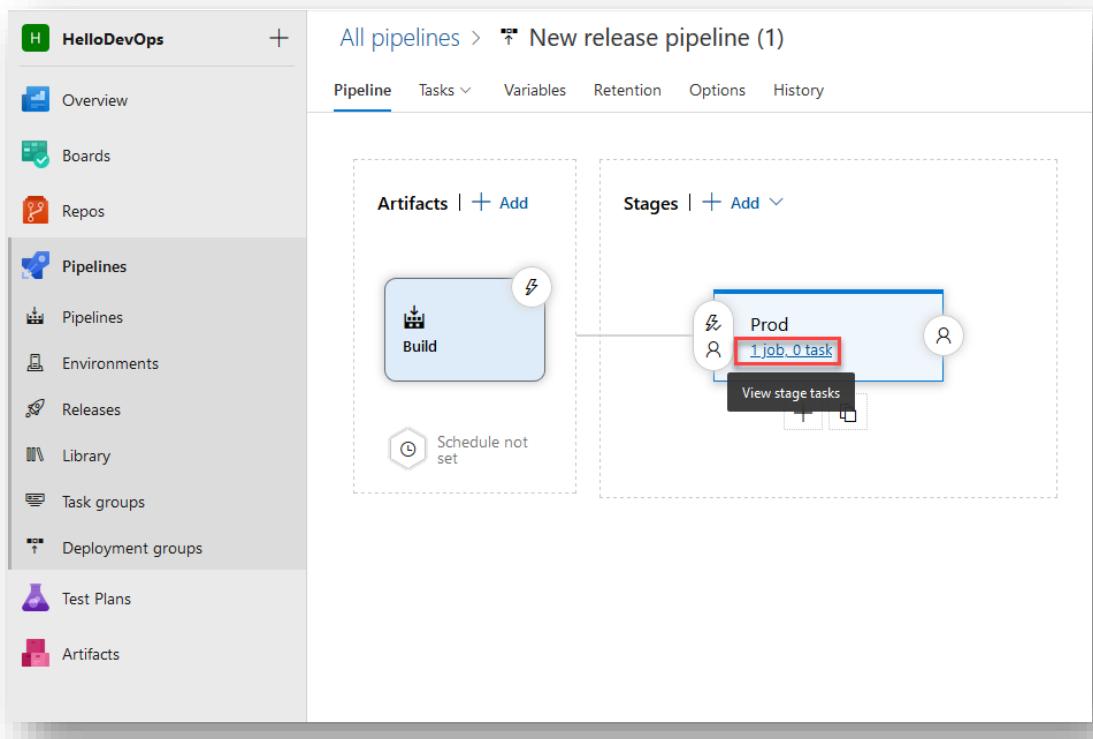
80. Give it a stage name and then close the **Stage** dialogue.



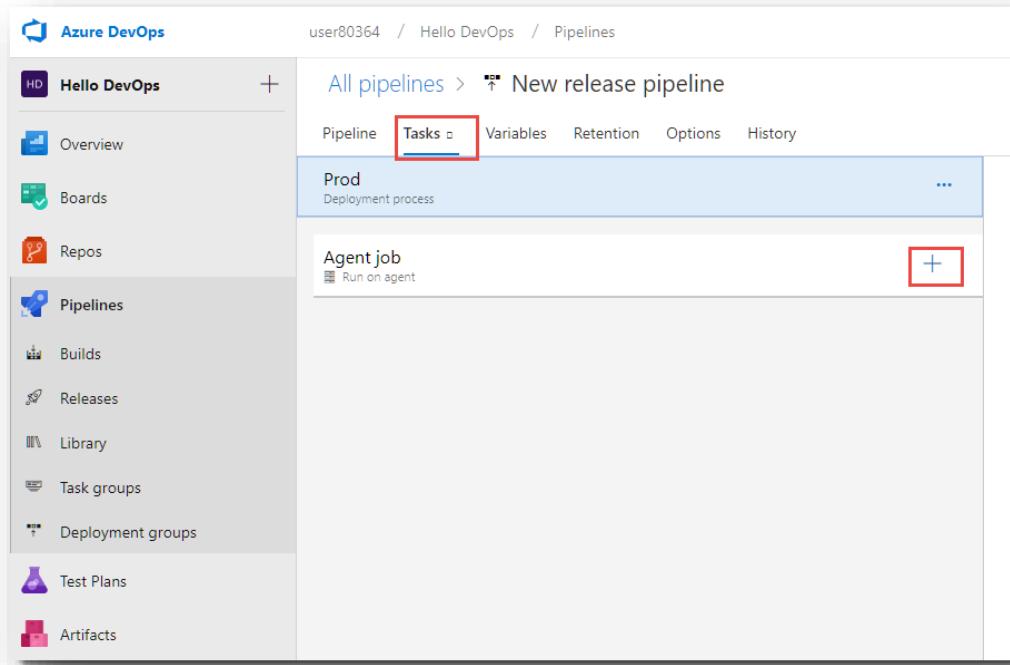
81. Next, we need to add the artifact that will be used in the deployment. Click the **Add** button in the **Artifacts** list. In the **Add an artifact** screen, select your “Build Managed Solution” from the source build pipeline. To make it easier to not have to deal with encoding spaces in filenames,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



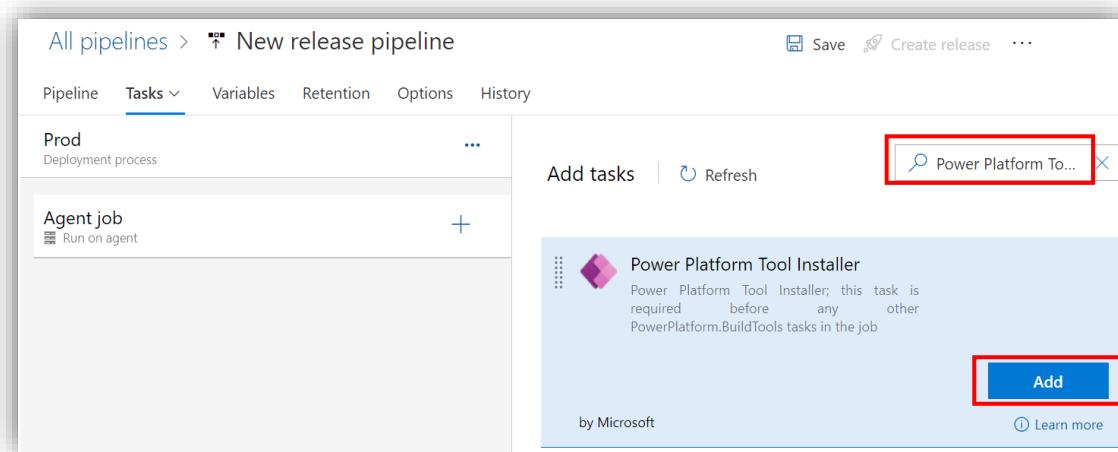
82. Switch to the **Tasks** view and click the plus + button to add a new task.

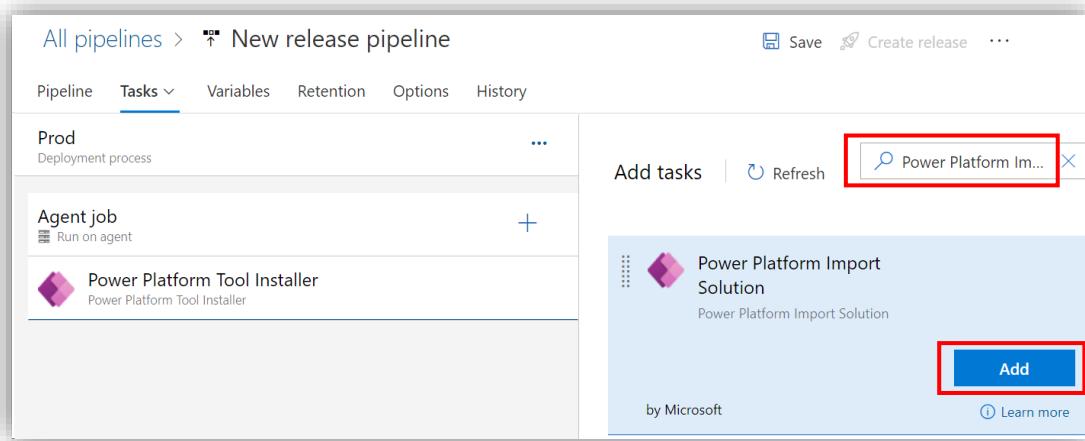
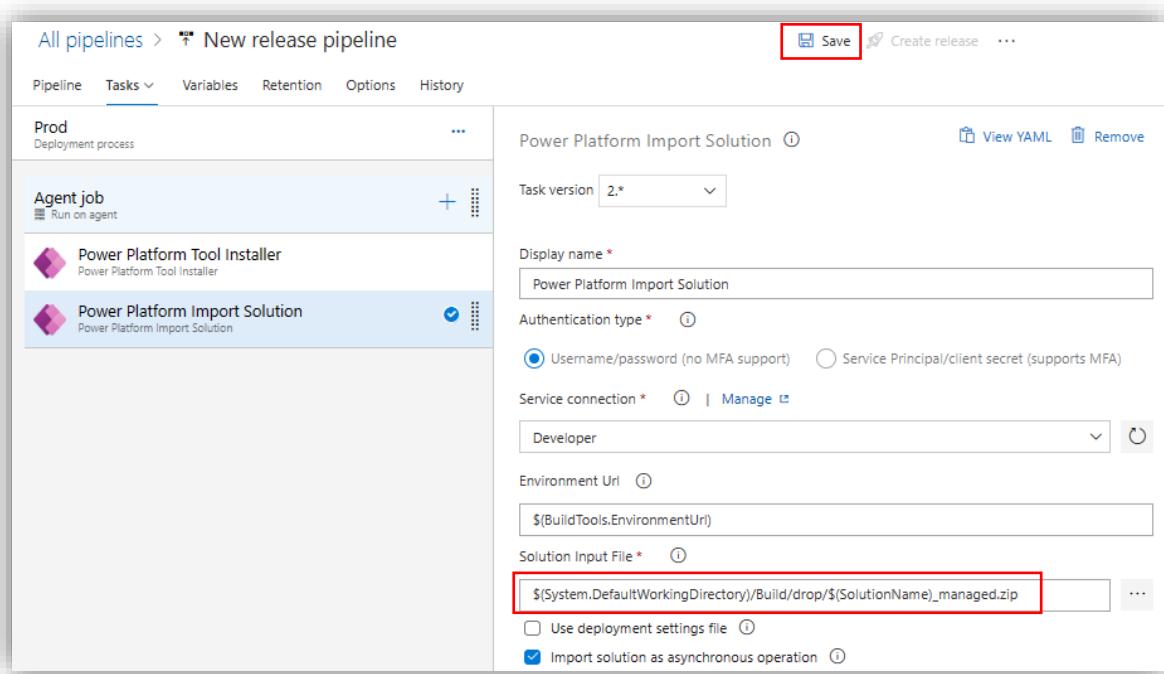


83. In the **Tasks** view, click the plus + button to add a new task.

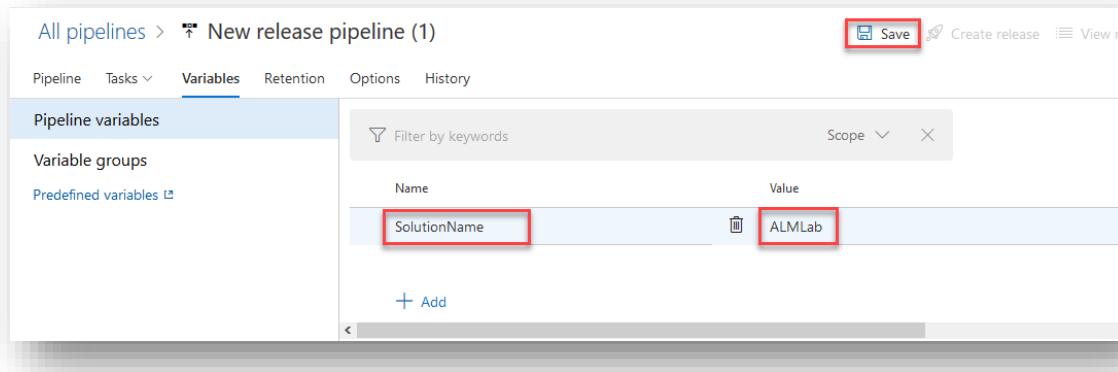


84. Add the **Power Platform Tool Installer** task.

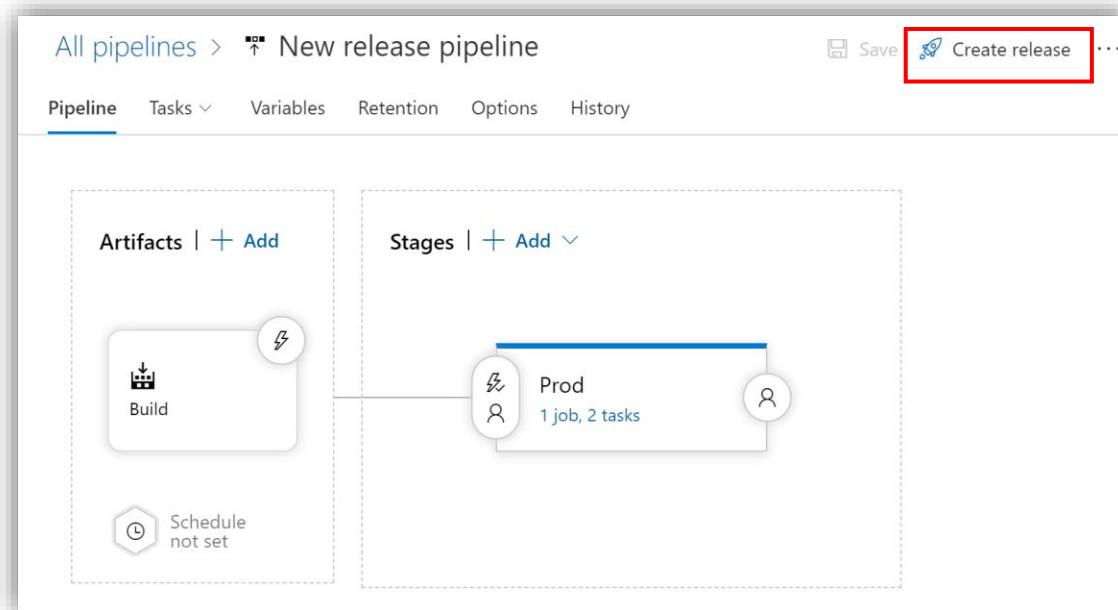


85. Add **Power Platform Import Solution** task to the pipeline.86. Configure the solution import to use the production environment connection and the following for solution input file:
\$(System.DefaultWorkingDirectory)/Build/drop/\$(SolutionName)_managed.zipClick **Save**.

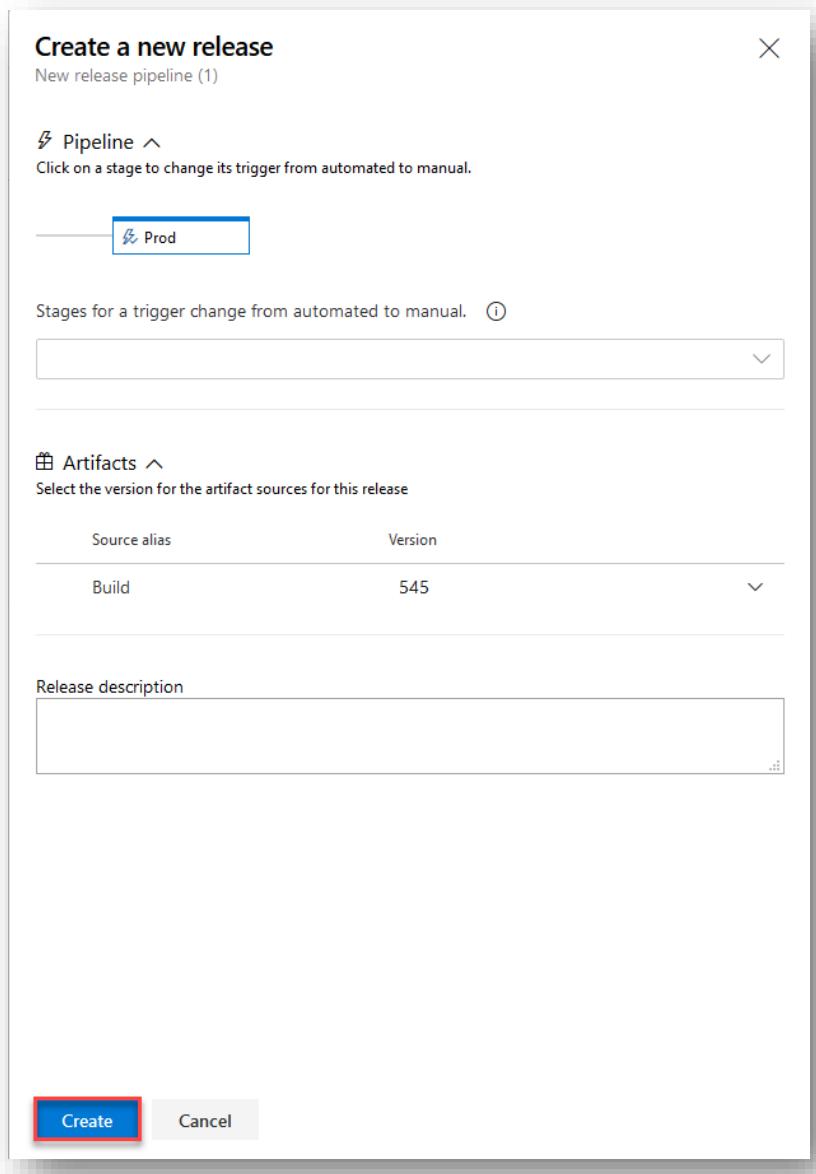
87. Switch to the **Variables** tab and add a variable named “SolutionName” and the unique name of the solution to import. If you are using the solution created in lab 1, the solution name should be “ALMLab”. Click **Save** and then **OK** in the next dialog.



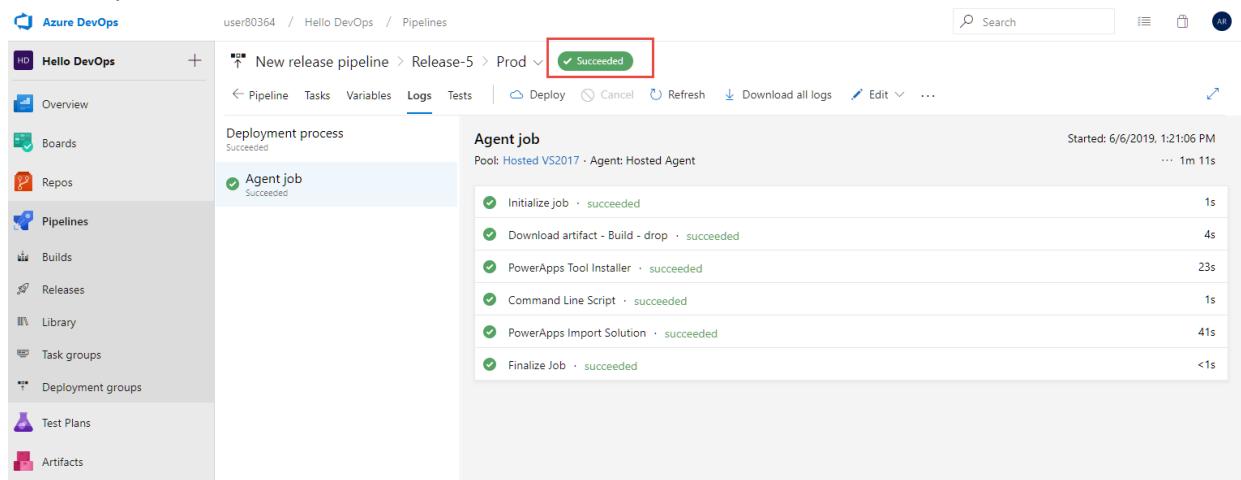
88. Click the **Create release** button. This functions like queuing a build pipeline, only for release pipelines.



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

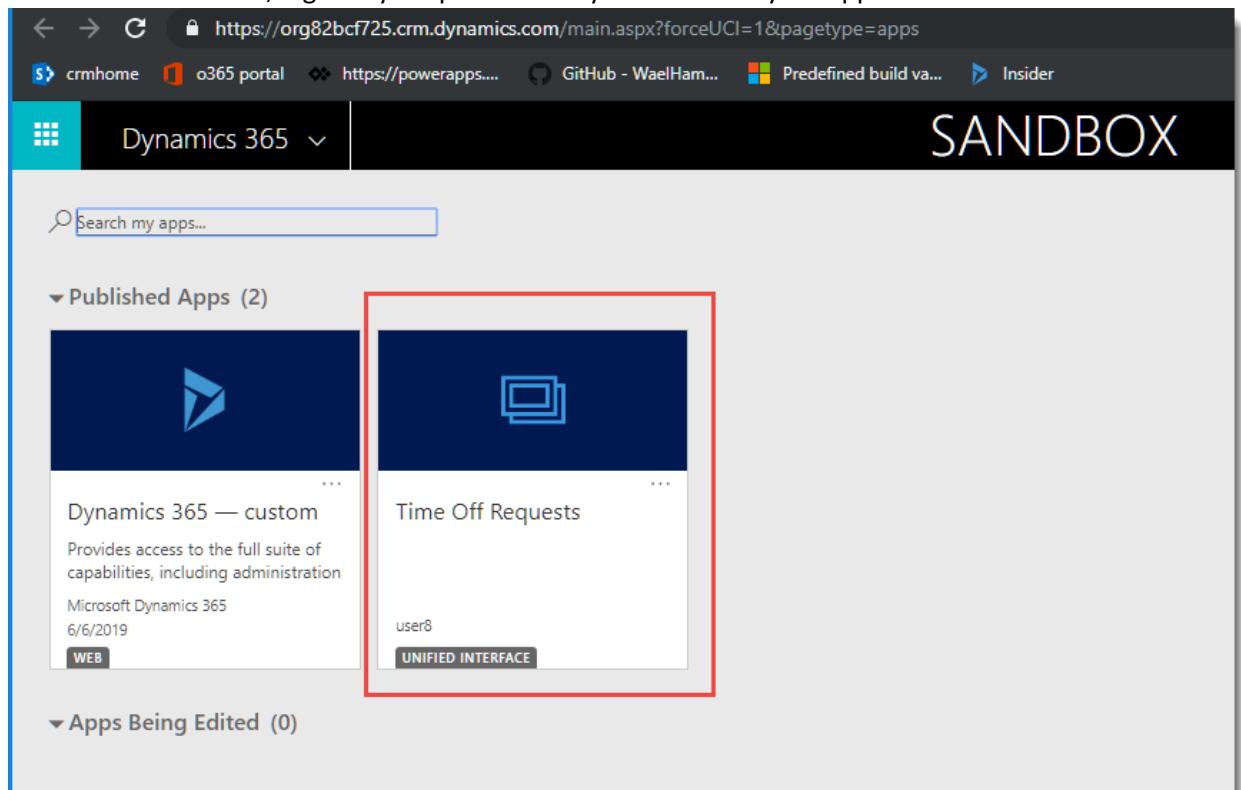


90. Monitor your release until it is succeeded or troubleshoot if it is not successful.



The screenshot shows the Azure DevOps interface for a 'Hello DevOps' project. On the left, the 'Pipelines' section is selected. In the center, a 'Release pipeline' named 'Release-5' is shown with a status of 'Succeeded'. A red box highlights the green checkmark icon and the word 'Succeeded' next to it. Below this, the 'Deployment process' is listed as 'Succeeded'. An 'Agent job' is detailed, showing tasks like 'Initialize job', 'Download artifact - Build - drop', 'PowerApps Tool Installer', 'Command Line Script', 'PowerApps Import Solution', and 'Finalize Job', all marked as 'succeeded'. The total duration of the job is 1m 11s.

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



The screenshot shows the Dynamics 365 app store interface. At the top, the URL is https://org82bcf725.crm.dynamics.com/main.aspx?forceUCI=1&pagetype=apps. The page title is 'Dynamics 365'. On the right, it says 'Sandbox'. Below the search bar, there's a section for 'Published Apps (2)'. One app, 'Dynamics 365 — custom', is listed. Another app, 'Time Off Requests', is listed and highlighted with a red box. This app has a blue icon, the name 'Time Off Requests', the user 'user8', and the label 'UNIFIED INTERFACE'. Below this, there's a section for 'Apps Being Edited (0)'.

Terms of Use

© 2020 Microsoft Corporation. All rights reserved.

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.