

ALM



# POWERPLATFORM

RELEASE PIPELINE



# DEPT – DYNAMICS 365/PORTALS ALM

## CI/CD – PIPELINE AUTOMATION EXTENSION

THIS DOCUMENT INFORMS THE READER ON THE PROCESS FOR SOURCE CODE VERSIONING FOR DYNAMICS 365 SOLUTIONS, REFERENCE DATA, AND POWERAPPS PORTALS USING AZURE DEVOPS POWERPLATFORM BUILD TOOLS, GIT, AND THE POWERPLATFORM CLI

# SUMMARY

The purpose of this document is to describe and illustrate the application lifecycle management (ALM) strategy at DEPT. The Dynamics 365 implementation at DEPT is quickly growing and new features are being added every quarter. In addition to new features, more and more line of businesses are onboarding as well. To accommodate frequent changes and optimizations to the platform to meet evolving business requirements, DEPT is embarking on an exercise to optimize their ALM process to achieve a greater level of configuration management integrity.

## ALM CHECKLIST

CONVENTIONS
Solutions are unpacked and version controlled in GIT hosted repo in Azure DevOps
Solutions are repacked through an automated process and stored as artefacts for releasees
Packages are built through an automated process that combines the required solutions and configuration data into a PackageDeployer.zip file
Solutions are exported and imported as managed
There is at least one discrete development instance per solution being developed (by line of business). A base solution includes common components is imported into LOB environments
Where multiple solutions are developed for deployment to a single production environment, the same solution publisher and prefix are used
Pipeline can support both standard solutions and solution patches
Solution.zip files and their contents are never manually edited
Where a solution being developed has dependencies on one or more solutions, the dependencies are satisfied through always importing those solutions as managed solutions into the discrete development instance for the solution being developed (e.g. Base solution)
Only managed solutions are deployed to environments downstream of development
All solutions are deployed via the release pipeline
No unmanaged changes are made directly to environments downstream of development
Each test case has traceability back to requirement
Test cases are automated
The pipeline supports configuration data transfers
Exported configuration data is saved as artefacts
The pipeline's runtime variables map to their own stage and run in the sequence they are provided by the user
If any stage fails, the pipeline exits (so will not execute subsequent stages to avoid faulty deployments)
Each stage generates a log artefact (e.g. Solution, Portal checker, diagnostics)
Release managers can create a release from a build which will leverage the artefacts from the build (instead of exporting these from an environment)
Any issues that are deemed critical by the solution checker and the portal checker will exit the pipeline
A default variable group is available to the build team. Each developer can clone the variable group for specific connection parameters. Alternatively, the organization can opt to leverage a single variable group to govern the connection parameters to each environment. Developers will always have access to a runtime variable and successful builds will store the resulting artefacts from these variables for future release.
The primary development stream solution set is static. Developers can create as many solutions as they need however, their components must be migrated to the primary solutions that make up the system. The same applies for the schema files. The pipeline allows a developer to deploy their solutions to the primary dev environment where the pipeline will merge its components to the primary solution (OPTIONAL) In a scenario where DEPT opts for developer environments or developer solutions, the pipeline will support solution component mover API calls to merge developer solution components to the BASE, CanExport, and Sonar 360 solutions (and others for future LOB implementations)

# TABLE OF CONTENTS

- DEPT – DYNAMICS 365/PORTALS ALM ..... 2
- CI/CD – PIPELINE AUTOMATION EXTENSION .....2
- Continuous IntegrationN.....5
  - Option 1: Developer invoked integration ..... 5
  - Option 2: Integration Automation ..... 5
  - Summary ..... 5
- Continuous Deployments.....6
  - Option 1: Release manager intervention ..... 6
  - Option 2: Deployment Automation ..... 6
- Security .....6
- Service Connections .....6
- Variables & Variable Groups .....7
  - Variable Group: Connection-Parameters..... 7
  - Runtime Variables..... 7
- .NET/NuGET Libraries / Marketplace (Dependencies) .....8
- Developer initiated multi-stage pipeline.....8
- Release Manager multi-stage release .....12
- CONFIGURING THE CI/CD PIPELINE.....16
  - Pre-Requisites ..... 16
  - Install the following marketplace plugins..... 17
  - Create Pipeline.....18
  - Create The Release Pipeline ..... 22

# IMPLEMENTATION

## OVERVIEW

The PowerPlatform CI/CD implementation is comprised of a scripts that provides the PowerPlatform build teams with the ability to check-in their Dataverse solutions, configuration data schema files, and portal code to the organizations primary repository(ies) hosting the source code and deployment artefacts that make up the Department's grant funding system. This pipeline extension is built to be generic to any PowerPlatform implementation that is comprised of both Model-Driven-Apps and PowerApps Portals. Canvas Apps are not yet fully supported in this implementation, however support for these types of applications will be released in version 1.3. That being said, since Canvas apps are typically stored in a solution, this pipeline automation extension can include Canvas Apps but will not include certain features like automated tests designed for these types of applications. Version 1.3 is scheduled to be released on June 28, 2022 along with an updated version of this document.

## CONTINUOUS INTEGRATION

For the organization to achieve stability, predictability and quality of their Dataverse application deployments, the build team needs to be equipped with tools that will automate the export for solutions, configuration data, and portal data and apply version control. In addition, members of the build team need the capability to deploy new features and or bug fixes frequently to minimize the complexities associated with less frequent and very large deployments. Having the ability to continuously release changes to the Dataverse applications will allow the developer to address issues such as missing dependencies, critical bugs, and potential misconfigurations or mis-interpreted requirements which can cause issues downstream. By catching these issues early, the team can resolve these issues and re-deploy. The deployment processes in the PowerPlatform can take a significant amount of time if executed manually which can cause overall productivity issues. Therefore, by implementing a robust continuous integration framework, developers can instead rely on the DevOps Pipelines feature to automate deployments which will not only provide overall productivity gains, but will also standardize the deployment processes, provide a facility for automated testing to ensure best practices are followed, validate that the solution, data and portal artefacts can be successfully deployed based on their latest changes etc. In this section, the extension for CI is described.

### OPTION 1: DEVELOPER INVOKED INTEGRATION

Developers have the option of manually invoking a pipeline for which the developer will be supply a series of variable values in a specific sequence. Once the variables are defined, and the pipeline invoked, the extension will interpret the supplied variables and will deploy the developer's solution(s), configuration data (from the schema), and (or) the portal configuration changes automatically. As part of this process, the extension will run the solution and portal checker API's provided by the Dataverse build tools and will export the Dataverse resources informed by the variable(s) provided by the developer. These artefacts are stored in the DevOps project storage for future usage such as release to a downstream environment such as UAT and PROD.

### OPTION 2: INTEGRATION AUTOMATION

The organization can configure the pipeline to trigger automatically via Pull-Requests. In this scenario, pull-requests are invoked to a branch such as TEST which will automatically deploy the solutions, data files, and (or) portals using pre-defined variable values in both the variable group and run time variables. In this scenario developers will be responsible to move their solution components to the primary solutions set in the variable artefacts. The release manager will then issue a PR to the relevant branch will execute the pipeline, save the artefacts in preparation for a release to a downstream environment. The benefits of this approach include a reduction of effort associated with developers having to manually invoke pipelines. The downside will be less traceability of who made the changes and when as with this approach, changes are published by a single (or multiple) release managers for the sole purpose of deployment and generation of artefacts to deploy to downstream environments.

## SUMMARY

DEPT can opt to use a combination of the two options instead of picking one. In this scenario, developers will be responsible to commit their work, indirectly through the pipeline script in DevOps, and the cadence for option 2 would be that the release manager would run the same pipeline at specific time intervals during a release cycle to prepare the artefacts for a full release to UAT and Production. If DEPT opts to leverage

only a single option, Option 1 would be more beneficial as it would align better with modern ALM standards whereby developers can test their own releases and receive feedback on any potential defects associated with their development artefacts through the automated test outputs provided by the pipeline so that when its time a full release, issues like missing dependencies, bugs and mis-configurations are minimized or eliminated resulting in a more reliable deployment process.

## CONTINUOUS DEPLOYMENTS

This pipeline extension includes a release pipeline targeted to UAT and PRODUCTION environments. Releases are invoked by release managers who will have the ability to select any successful and issue a release to UAT and PROD using that build’s artefacts. The script is identical to the build integration script and will use the same variable group. The only delta is that the release pipeline script will loop through its source build integration artifacts and set the runtime variables automatically so that the release manager does not need to re-set these variables at run time.

### OPTION 1: RELEASE MANAGER INTERVENTION

The release pipeline can be invoked manually by a release manager from any build or from the release’s menu in DevOps.

### OPTION 2: DEPLOYMENT AUTOMATION

Like the automated build integration, a release manager can choose to have the release pipeline execute as part of a PR and thus tied to a branch. The processes demonstrated below are the same, with t

# DESIGN

## SECURITY

Only build administrators can modify the pipeline and execute the full release pipeline. Contributors (developers) can execute the build integration pipeline. This baseline implementation also restricts any modifications to the variable group that is linked to the pipelines. Only build administrators can modify these groups. However if DEPT employs a developer environment strategy, it is recommended that the variable groups are cloned and that each developer will have access to the build integration pipeline within their own branch and these would be tied to a variable group whose source and target would reflect their developer environment and point to the consolidated dev environment.

This current version of the scripts supports OOB variable groups and variable secrets but assumes that the DevOps projects is not linked to a KeyVaults. If DEPT links the project to a KeyVaults, the variable group values will be updated to include the KeyVaults name and secret IDs instead of using “masked” secrets.

## SERVICE CONNECTIONS

The table below includes the service connections required for this extension. In addition to service connections, since this extension is leveraging the community version of the build tools, connecting strings must also be configured. However, the connection strings are automatically generated by other variables (e.g. the SPN – App Registrations/App User’s TenantID, URL, ClientID and Secret all of which are concatenated within the source and target connection strings for each environment. These variables, including the service connections (Target and SourceSPN-{ENV} variables are available in the Variable Group (Connection-Parameters).

A service connection should be created for each environment that will service as a source and target environment for the integration and deployment pipelines. All connection parameters should be pre-defined in the connection parameters variable group.

## VARIABLES & VARIABLE GROUPS

The table below lists all variables and groups and describes their purpose and usage.

### VARIABLE GROUP: CONNECTION-PARAMETERS

Variable Name	Description
<b>ClientID</b>	App Registration's ClientID (ApplicationID – SPN for Dataverse Environments' App User (assumes 1 SPN for multiple environments)
<b>DeploymentProfile-PROD</b>	Portals -> PROD Global settings (located in Deployment-Profiles folder in repository – prod.deployment.yml)
<b>DeploymentProfile-TEST</b>	Portals -> TEST Global settings (located in Deployment-Profiles folder in repository – test.deployment.yml)
<b>DeploymentProfile-UAT</b>	Portals -> UAT Global settings (located in Deployment-Profiles folder in repository – uat.deployment.yml)
<b>Secret</b>	App Registration's Client Secret (SPN for Dataverse Environments' App User (assumes 1 SPN for multiple environments)
<b>SourceConnection-DEV</b>	Value is populated by ClientID, Secret, Source and Target URLs, and TenantID – this is for data transfers
<b>SourceConnection-UAT</b>	Value is populated by ClientID, Secret, Source and Target URLs, and TenantID – this is for data transfers
<b>SourceSPN-DEV</b>	Service Connection name for DEV (Source environment)
<b>SourceSPN-UAT</b>	Service Connection name for UAT (Source environment)
<b>SourceURL-DEV</b>	Source URL for DEV (feeds SourceConnection variable)
<b>SourceURL-UAT</b>	Source URL for UAT (feeds SourceConnection variable)
<b>TargetConnection-DEV</b>	Value is populated by ClientID, Secret, Source and Target URLs, and TenantID – this is for data transfers (source environment connection string)
<b>TargetConnection-UAT</b>	Value is populated by ClientID, Secret, Source and Target URLs, and TenantID – this is for data transfers (target environment connection string)
<b>TargetSPN-DEV</b>	Service Connection name for DEV (Target environment)
<b>TargetSPN-UAT</b>	Service Connection name for UAT (Target environment)
<b>TargetURL-PROD</b>	Target URL for PROD (feeds TargetConnection variable)
<b>TargetURL-UAT</b>	Target URL for PROD (feeds TargetConnection variable)
<b>TargetURL-DEV</b>	Target URL for DEV (feeds TargetConnection variable)
<b>TenantID</b>	Azure tenant ID hosting App registration record

### RUNTIME VARIABLES

Variable Name	Description
<b>Artefact1</b>	Provide the name of a solution (doesn't need to be in the repository, the pipeline will perform the export and commit etc.) or configuration migration xml file (must be in the data directory in the repo)
<b>Artefact1-Target-Solution</b>	If Artefact1 is a solution, and you need to copy its component to a primary solution, provide the name of the primary solution here
<b>Artefact2</b>	Provide the name of a solution (doesn't need to be in the repository, the pipeline will perform the export and commit etc.) or configuration migration xml file (must be in the data directory in the repo)
<b>Artefact2-Target-Solution</b>	If Artefact2 is a solution, and you need to copy its component to a primary solution, provide the name of the primary solution here
<b>Artefact3</b>	Provide the name of a solution (doesn't need to be in the repository, the pipeline will perform the export and commit etc.) or configuration migration xml file (must be in the data directory in the repo)
<b>Artefact3-Target-Solution</b>	If Artefact3 is a solution, and you need to copy its component to a primary solution, provide the name of the primary solution here

<b>Artefact4</b>	Provide the name of a solution (doesn't need to be in the repository, the pipeline will perform the export and commit etc.) or configuration migration xml file (must be in the data directory in the repo)
<b>Artefact4-Target-Solution</b>	If Artefact4 is a solution, and you need to copy its component to a primary solution, provide the name of the primary solution here
<b>Deploy Portal?</b>	If set to yes, the pipeline will export the portal via CLI and commit to source. It will also deploy the portal to the target environment (using SourceURL and TargetURL)
<b>Comments</b>	Used to issue the git comments when the pipelines automatically commit your artefacts
<b>Project Name</b>	Optional, provides project context
<b>Variable-Group</b>	Name of the variable group that is linked to the pipeline to leverage. The group must have the same variable names (and types) as the "Connection-Parameters" variable group. This is useful to isolated dev environments with specific Dataverse environments that fall outside the main dev stream.

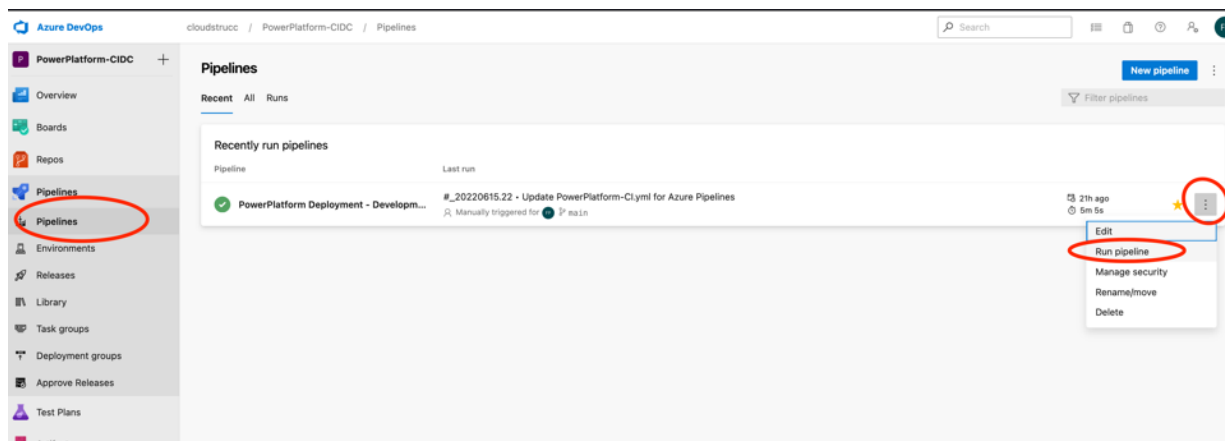
## .NET/NUGET LIBRARIES / MARKETPLACE (DEPENDENCIES)

The following marketplace plugins and nuget packages are leveraged in this extension. The goal is to reduce the number of marketplace plugins and slowly transition towards using the CLI exclusively coupled with Dataverse API calls to reduce dependencies.

Name	Type	Purpose
<b>PowerPlatform Build Tools</b>	DevOps Marketplace Plugin	Helper YAML extensions for pipelines
<b>Power DevOps Tools</b>	DevOps Marketplace Plugin	Helper YAML extensions for pipelines
<b>PowerPlatform CLI</b>	Nuget Package	Office Microsoft CLI for PowerPlatform

## DEVELOPER INITIATED MULTI-STAGE PIPELINE

The process below illustrates the typical process for which a developer will execute the build integration pipeline. In this example, the developer is transferring Organizations and Contacts from developed to a staging environment before transferring the CanExport solution as this solution has workflow dependencies relying on Organization and Contact lookup values being populated. At the same time, the user wants to deploy his/her latest portal changes to the staging. \*NOTE\* to enable CI, you can opt to have this pipeline trigger when a PR issued to a specific branch. However, the process below is recommended as it provides the build team with the ability to check-in their work and automatically deploy and commit their changes and generate the artefacts and test logs. If you decide to employ full automation, simply update the trigger of this pipeline to a specific branch and ensure that the runtime variables configured are pre-defined (see variables section) – the variable group is already pre-defined so not additional configuration required there.





Run pipeline

×

Select parameters below and manually run the pipeline

branch/tag

development

▼

Select the branch, commit, or tag

Advanced options

Variables

4 variables defined

>

Stages to run

Run as configured

>

Resources

Use latest version of all resources

>

←

Variables

×

PowerPlatform Deployment - Development & Test

<div>fx</div>	Artefact1 = CanExport	>
<div>fx</div>	Artefact2 = Stakeholders.xml	>
<div>fx</div>	Artefact3 = CanExportAutomation	>
<div>fx</div>	Artefact4 =	>
<div>fx</div>	Comment = New CanExport Application Fields Stories: #321, 435	>
<div>fx</div>	Deploy Portal? = Yes	>
<div>fx</div>	Project Name =	>
<div>fx</div>	Variable Group = Connection-Parameters	>

## Run pipeline




Select parameters below and manually run the pipeline

### Branch/tag

 development 

Select the branch, commit, or tag

### Advanced options

**Variables**   
8 variables defined

**Stages to run**   
Run as configured

**Resources**   
Use latest version of all resources

☐ Enable system diagnostics

Cancel

Run

**#\_20220616.3 Update PowerPlatform-CI.yml for Azure Pipelines** Run new

PowerPlatform Deployment - Development & Test

This run is being retained as one of 3 recent runs by main (Branch). View retention leases

**Summary** Releases Extensions

Manually run by **Frederick Pearson**

Repository and version  
PowerPlatform-CIDC  
main 5cf7ba82

Time started and elapsed  
Today at 9:40 p.m.  
21m 14s

Related  
0 work items  
5 published

Tests and coverage  
[Get started](#)

**Warnings** 4

- The names of some imported commands from the module 'Microsoft.Xrm.WebApi.PowerShell' include unapproved verbs that might make them less discoverable. To find the commands with unapproved verbs, run th...
- The names of some imported commands from the module 'Microsoft.Xrm.WebApi.PowerShell' include unapproved verbs that might make them less discoverable. To find the commands with unapproved verbs, run th...
- The names of some imported commands from the module 'Microsoft.Xrm.WebApi.PowerShell' include unapproved verbs that might make them less discoverable. To find the commands with unapproved verbs, run th...
- The names of some imported commands from the module 'Microsoft.Xrm.WebApi.PowerShell' include unapproved verbs that might make them less discoverable. To find the commands with unapproved verbs, run th...

**Stages** Jobs

<b>Artefact1</b> 2 jobs completed 3m 25s 2 artifacts	<b>Artefact2</b> 2 jobs completed 1m 53s 1 artifact	<b>Artefact3</b> 2 jobs completed 2m 31s 1 artifact	<b>Artefact4</b> Skipped	<b>Portals</b> 1 job completed 8m 1s 1 artifact
---	--	--	-----------------------------	--

If successful, each stage (1 per artefact specified in the variables), will store its artefact in the Agent's Artefact Staging Directory for a release to a UAT and eventually PROD environment. The pipeline execution's success means that the source developer environment artefacts provided in the variables will be deployed on the target staging environment (or test environment, depending on your deployment strategy for build integration). The solutions are deployed as managed, and the portal is deployed using the CLI and only delta portal changes are also deployed to the target staging environment (or build environment).

To view the artefacts, click on the artefact anchor in one of the stages. You can then view the download the artefacts and the solution and portal checker log files for review. Note that this is the happy path. If any of the stages fail, the pipeline will exit to ensure that no subsequent solutions, data files or portal is deployed. In this case, you can click on the failed stage(s) to view the detailed log of the failure.

Below demonstrates the artefact storage structure. Notice that the folder names will include an integer as a suffix which informs the eventual release of the sequence for which to deploy these artefacts to downstream environments.

*\*Note that as you add additional tests to this CI/CD extension, you should consider adding your test report(s) to the artefacts instead of just in the GIT repository so that in the future, you have the ability to leverage other pipeline tasks to send these report artifacts to a SIEM to archiving system\**

**Artifacts**

**Published**

Name

- Data2
  - Data.zip
- Portal
  - customer-self-service
  - Deployment-Profiles
  - README.md
- Sin
  - CanExport.zip
  - PowerAppsChecker
- Sin3
  - Base.zip
  - PowerAppsChecker
- SolutionAnalysisLogs
  - 20220616014231\_CanExport.sarif
  - 20220616015236\_Base.sarif
  - IssueResultSummary.md

In the event the pipeline execution fails, you can view the logs by clicking on any failed stage to view the details. In the example below, the TargetSPN was misspelled in the variable group resulting in a connection issue to the target environment for the data import.

StagesJobs

Artefact1

2 jobs completed2m 22s

2 artifacts

Artefact2

2 jobs completed2m 13s

1 artifact

Artefact3

2 jobs completed3m 2s

1 artifact

Artefact4

Failed38s

Artefact4SIn

Artefact4Data

Rerun failed jobs

Rerun all jobs

Portals

Skipped

Jobs in run #\_20220615.19

Artefact2

Artefact2SIn2m 10s

Artefact2Data

Artefact3

Artefact3SIn2m 59s

Artefact3Data

Artefact4

Artefact4SIn

Artefact4Data35s

Initialize job2s

Transfer your data...2s

Checkout PowerPla...7s

MSCRMToolInstaller3s

MSCRMPing3s

MSCRMExportCM...12s

MSCRMImportCMD...3s

Publish and Artifa...<1s

Post-job: Checkou...<1s

Finalize Job<1s

MSCRMImportCMDData

Starting: MSCRMImportCMDData

Task : Import Config Migration Data

Description : Import data exported using Configuration Migration Tool into a CRM instance

Version : 12.0.4

Author : Wael Hamze

Help : More information on Configuration Migration Tools can be found [here](https://docs.microsoft.com/en-us/power-platform/admin/manage-configuration-dat

Download Microsoft.Xrm.Tooling.ConfigurationMigration 1.0.0.64 to: C:\hostedtoolcache\windows\MSCRMBuildTools\12.0.66

Download Microsoft.Xrm.Tooling.CrmConnector.PowerShell 3.3.0.928 to: C:\hostedtoolcache\windows\MSCRMBuildTools\12.0.66

Microsoft.Xrm.Tooling.CrmConnector.PowerShell 3.3.0.928 already cached in C:\hostedtoolcache\windows\MSCRMBuildTools\12.0.66\Microsoft.Xrm.Tooling.CrmConnector.Pow

C:\hostedtoolcache\windows\MSCRMBuildTools\12.0.66\Microsoft.Xrm.Tooling.CrmConnector.PowerShell.3.3.0.928

##[error]Failed to connect to CRM. One or more errors occurred. => An error occurred while sending the request. => The remote name could not be resolved: 'goc-theme

One or more errors occurred. => An error occurred while sending the request. => The remote name could not be resolved: 'goc-theme-release.crm3.dynamics.com'One or

One or more errors occurred. => An error occurred while sending the request. => The remote name could not be resolved: 'goc-theme-release.crm3.dynamics.com'Unable

Unable to Login to Dynamics CRM

##[error]Failed to connect to CRM. One or more errors occurred. => An error occurred while sending the request. => The remote name could not be resolved: 'goc-theme

One or more errors occurred. => An error occurred while sending the request. => The remote name could not be resolved: 'goc-theme-release.crm3.dynamics.com'One or

One or more errors occurred. => An error occurred while sending the request. => The remote name could not be resolved: 'goc-theme-release.crm3.dynamics.com'Unable

Unable to Login to Dynamics CRM

##[error]Cannot bind argument to parameter 'CrmConnection' because it is null.

Finishing: MSCRMImportCMDData

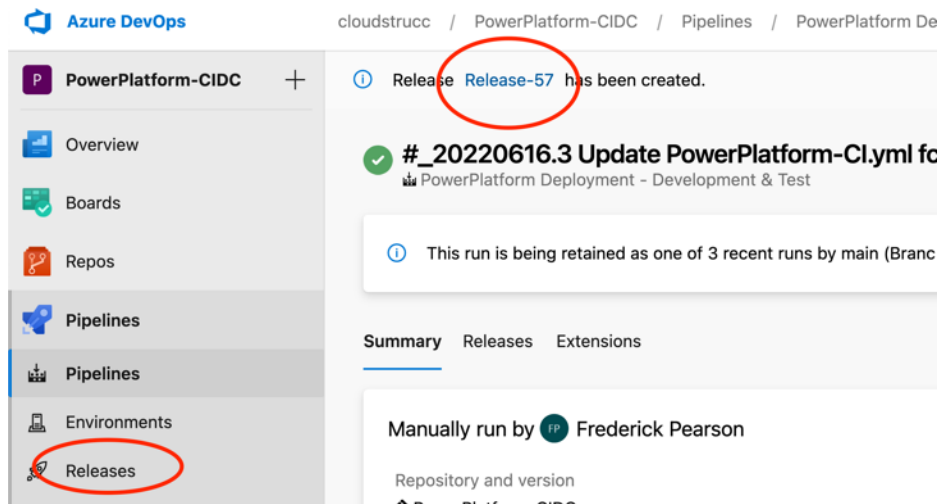
RELEASE MANAGER MULTI-STAGE RELEASE

The process below illustrates the typical process for which a release manager will execute a full release to UAT and invoke the PROD release manually only once UAT is completed. This is useful because, once the artefacts are deployed to UAT, the client can test against the acceptance criteria and if everything is ok, the release manager can return to this release and execute the production stage knowing that the same artefacts deployed to UAT successfully will be used to deploy to production.

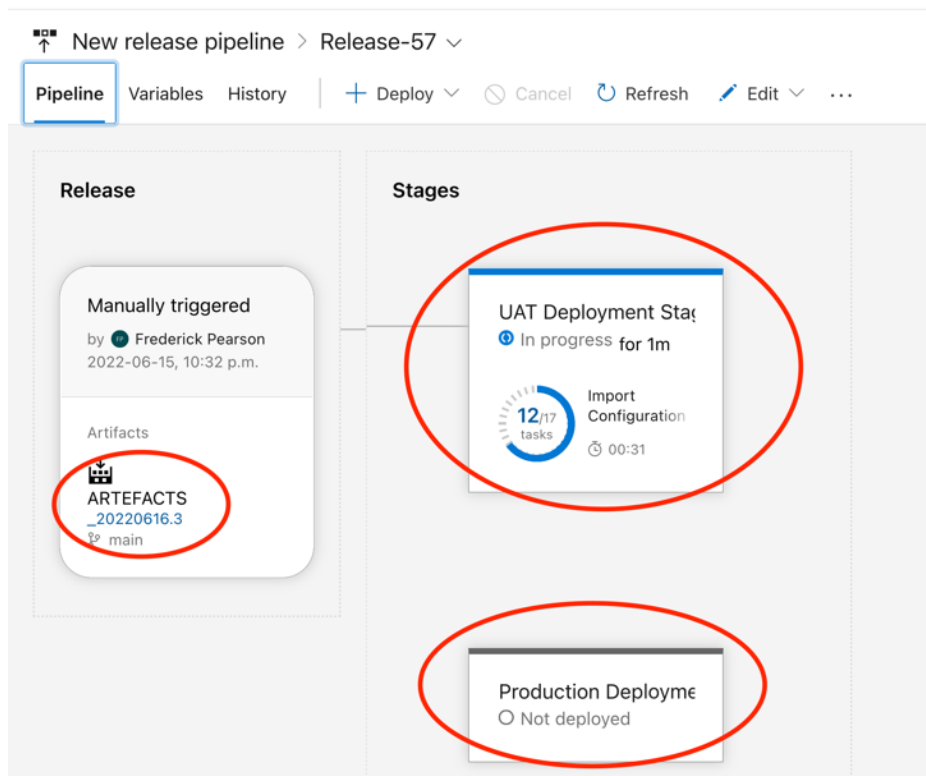
Using the example build from the previous section, the release manager can navigate to the successful build, select the ellipsis and press “Release”



Once created, the release anchor will be displayed just below the breadcrumb, but you can also view the progress of the release under the releases menu.



In the release details, you will notice that the UAT deployment is being executed whereas Production is not being deployed. In addition, the artefacts from the release are those that were generated by the successful build.



Once succeeded you should see the following.

*\*Note that if you would like to re-deploy the same artefacts but to a different environment, the UAT stage below relies on the same variable group used in the build integration pipeline. To do so you could update the TargetURL-UAT, TargetSPN-UAT, ClientID and Secret (if the App User is different in the new target) OR simply clone the Connection-Parameters variable group, re-run the same build (in this case \_20220616.3) and edit the Variable-Group variable to your new group name and re-run the build integration pipeline and issue a new release. The latter would be recommended to avoid any disruption from updating connection parameters of your main development stream. IN summary, if you would like to employ the same release pipeline deployment strategy to another set of downstream environment(s), simply clone the Connection-Parameters variable group, everything else an remain the same as this CI/CD extension allows you to specify a variable group at runtime to give you control over which environments you would like to apply CI/CD to\**

New release pipeline > Release-57

Pipeline Variables History + Deploy Cancel Refresh Edit

### Release

Manually triggered  
by Frederick Pearson  
2022-06-15, 10:32 p.m.

Artifacts

ARTEFACTS  
\_20220616.3  
main

### Stages

UAT Deployment Stage

Succeeded

1 warning  
on 2022-06-15, 10:36 p.m.

Production Deployment

Not deployed

Once UAT has been successfully completed, and you are ready to deploy to production. Navigate to the release that deployed the UAT artefacts that resulting in the successful UAT testing cycle and execute the production deployment

PowerPlatform-CIDC

- Overview
- Boards
- Repos
- Pipelines
  - Pipelines
  - Environments
  - Releases
- Library

Search all pipelines

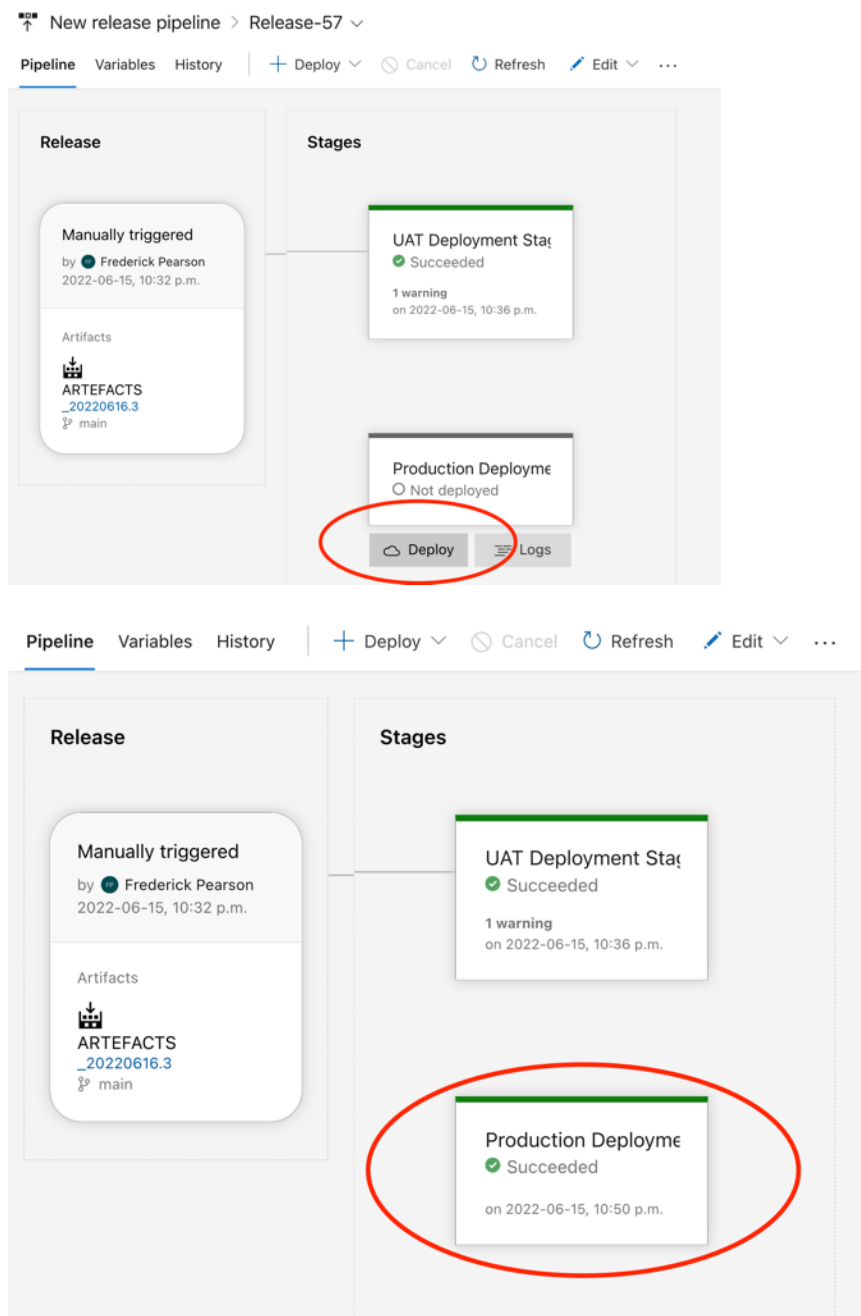
New release pipeline

UAT Deployment Stage

### New release pipeline

Releases Deployments Analytics

Releases	Created	Stages
<div>FP</div> <p><b>Release-57</b></p> <p>_20220616.3 main</p>	2022-06-15, 10:32:48 p.m.	<div>UAT Depl...</div>
<div>FP</div> <p><b>Release-56</b></p> <p>_20220615... main</p>	2022-06-15, 12:30:35 a.m.	<div>UAT Depl...</div>
<div>FP</div> <p><b>Release-55</b></p> <p>_20220603.7 main</p>	2022-06-14, 5:58:41 p.m.	<div>UAT Depl...</div>
<div>FP</div> <p><b>Release-54</b></p> <p>_20220603.7 main</p>	2022-06-14, 3:37:45 p.m.	<div>UAT Depl...</div>



## CONFIGURING THE CI/CD PIPELINE

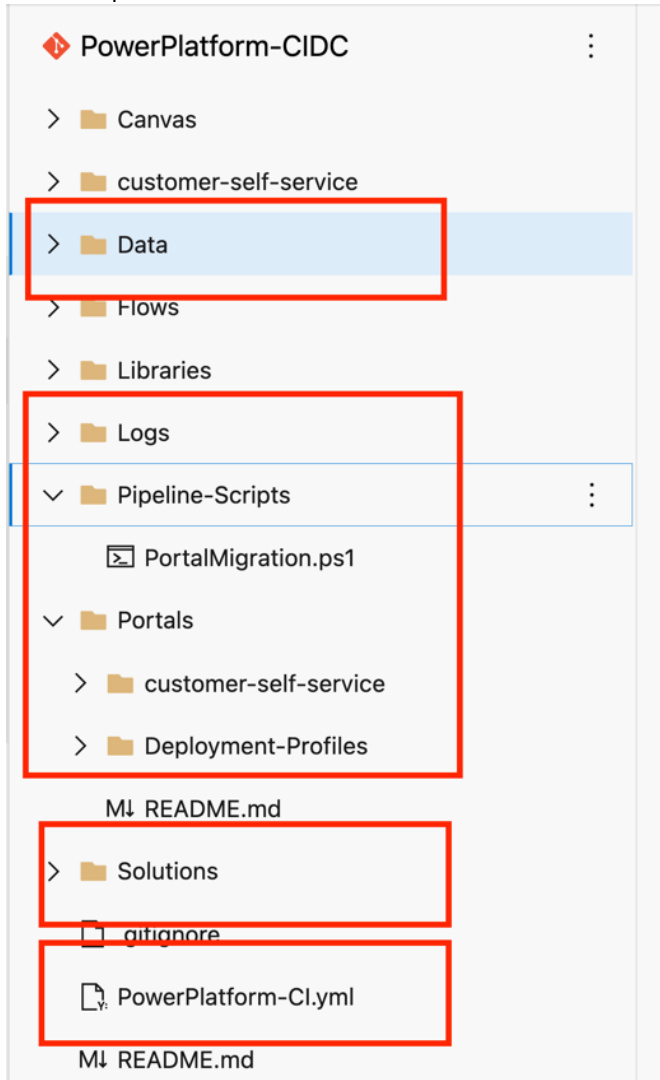
This section describes how to deploy and configure the CI/CD artefacts to your DevOps project.

### PRE-REQUISITES

- Commit the following [Pipeline Automation YAML File](#) to your repository's root directory
- Commit the following [PowerShell script](#) to a folder named "Pipeline-Scripts" in your repository's root directory
- Download the [release-pipeline JSON file](#)
- Create a Solutions folder in your repository's root directory
- Create a Data folder in your repository's root directory
- Create a ExportedData folder in your repository's Data folder (created in the previous step)
- Create a Portals folder in your repository's root directory
- Create a Logs folder in your repository's root directory
- Create a Deployment-Profiles folder in your repository's Profile directory
- Create a folder that matches the name of your PowerApps Portals website record in your repository's Portals folder. In that folder, create a deployment-profiles folder and include your dev, test, uat, and prod YAML profile files (e.g. test.deployment.yml <- where test is the CLI parameter flag that will inform the CLI which deployment profile to use for your portal deployments.














The example below demonstrates the above folder structure implemented in a DevOps repository to leverages this extension



INSTALL THE FOLLOWING MARKETPLACE PLUGINS

Extensions

Installed Requested Shared

-  **Azure AD B2C Release Tools** by DarkLoop  
A set of release tasks for Azure AD B2C
-  **Azure Pipelines Approval** by Gustavo Bergamim  
A simple way for view and approve releases
-  **Code Search** by Microsoft  
Code Search provides fast, flexible and accurate search across all your code
-  **Delivery Plans** by Microsoft  
Manage your portfolio of work with a calendar based view across teams and projects.
-  **HCL AppScan** by HCL Technologies  
Performs static, dynamic and open source security tests for your applications
-  **Personas** by Agile Extensions  
Create personas easily and map them to work items via tags.
-  **Power DevOps Tools** by Wael Hamze  
Tasks for automating Build & Deployment of Dynamics 365 CE/CDS/PowerApps Solutions
-  **Power Platform Build Tools (1.0.77)** by Microsoft  
Automate common build and deployment tasks related to Power Platform
-  **Run Inline Powershell and Azure Powershell** by Peter Groenewegen  
Build extension that enable you to run Powershell and Azure Powershell from a textbox.
-  **Send Email** by Rene van Osnabrugge (Xpirit)  
Send Email Build and Release Task
-  **Variable Toolbox** by Jesse Houwing  
Azure Pipelines tasks for setting and expanding variables.

## CREATE PIPELINE

This section demonstrates the steps to create the build integration pipeline.

### Create the Pipeline

Azure DevOps

cloudstrucc / PowerPlatform-CIDC / Pipelines

Search


**New pipeline**

Pipelines

Recent All Runs

Filter pipelines







Recently run pipelines

Pipeline	Last run
 <b>PowerPlatform Deployment - Developm...</b>	#_20220616.3 - Update PowerPlatform-CI.yml for Azure Pipelines Manually triggered for main 1h ago 21m 14s

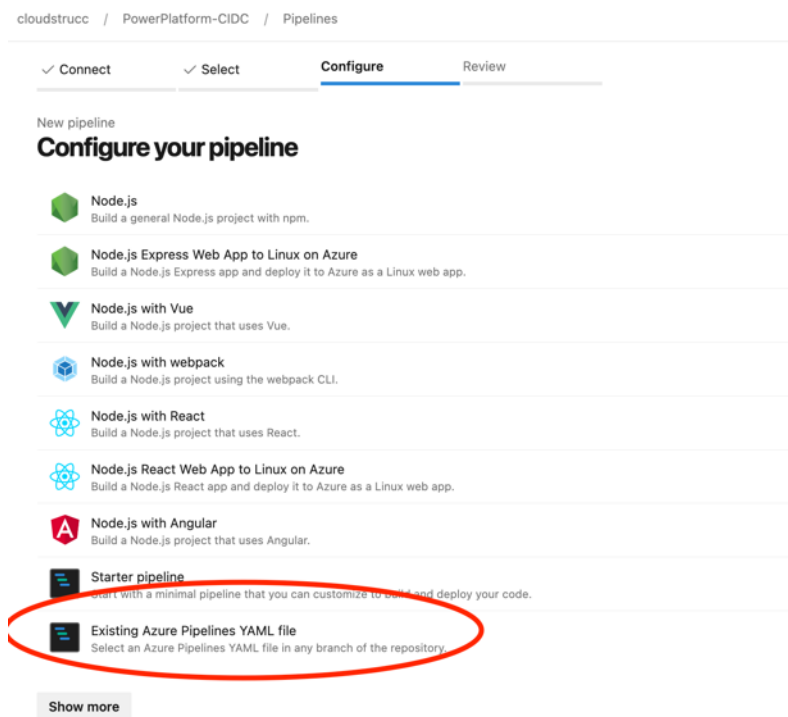
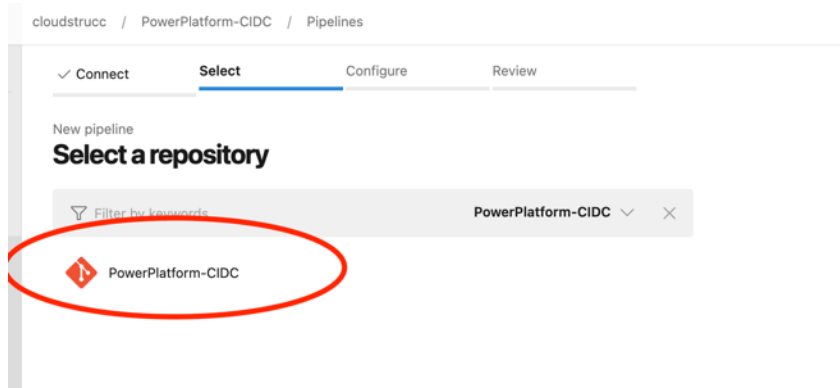
Connect Select Configure Review

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 generic Git repository
-  **Subversion**  
Centralized version control by Apache

Use the classic editor to create a pipeline without YAML.



Select the CI/CD YAML File provided for this extension, which needs to be hosted in your repository's root directory. You can also choose the branch for which the YAML file exists.

**Select an existing YAML file**

Select an Azure Pipelines YAML file in any branch of the repository.

Branch  
main

Path  
/PowerPlatform-CLYml

Select a file from the dropdown or type in the path to your file

PowerPlatform-CIDC

Cancel Continue

✓ Connect ✓ Select ✓ Configure **Review**

New pipeline

## Review your pipeline YAML

PowerPlatform-CIDC / PowerPlatform-CLYml

Show assistant

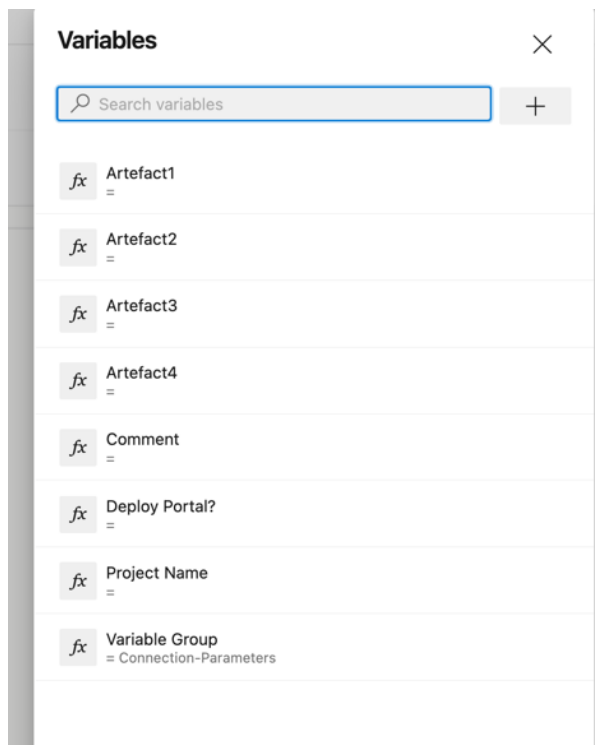
```

1 # PowerPlatform Continuous Integration Pipeline (Solution & Schema Files)
2
3 name: $(Project Name)_$(Date:yyyyMMdd)$(Rev:.r)
4 trigger:
5   - test
6
7 variables:
8   - Artefact1IsData: $[contains(variables['Artefact1'], 'xml')]
9   - Artefact2IsData: $[contains(variables['Artefact2'], 'xml')]
10  - Artefact3IsData: $[contains(variables['Artefact3'], 'xml')]
11  - Artefact4IsData: $[contains(variables['Artefact4'], 'xml')]
12  - DeployPortal: $[contains(variables['Deploy Portal?'], 'Yes')]
13  - ExitPipeline: false
14  - group: '$(Variable Group)'
15
16 pool:
17   vmImage: windows-latest
18
19 stages:
20   - stage: Artefact1
21     condition: ne(variables.Artefact1, '')

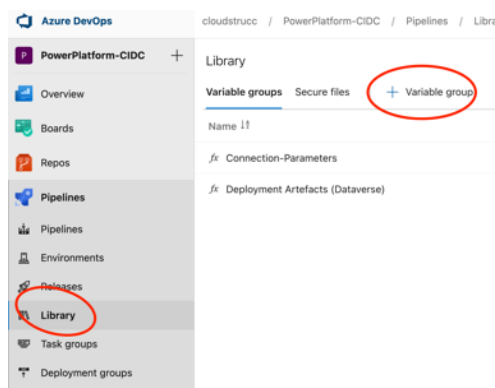
```

Variables Run

Next create all the following variables and make sure to set the variable group name to “Connection-Parameters” or whichever name you will use for your variable group you will create in the next section. Once completed, press the drop down next to the blue run button and press Save **(Do not attempt to run the pipeline at this point)**



### Create Variable Group & Link to Pipeline



When creating the group ensure that the variable names match those listed in the [Variable Group section of this document](#)

Library > Connection-Parameters

Variable group

Save Clone Security Pipeline permissions Approvals and checks

**Properties**

Variable group name

Connection-Parameters

Description

Variables holding connection parameters to the Dataaverse for both the first party Microsoft PowerPlatform build tools and the community PowerPlatform build tools for Development and Test environments

☐ Link secrets from an Azure key vault as variables

**Variables**

Name	Value
ClientID	4ff994b1-9789-40e1-874c-50fc92007812
DeploymentProfile-PROD	prod
DeploymentProfile-TEST	release
DeploymentProfile-UAT	uat

The pipeline is not fully configured and can be executed at anytime

## CREATE THE RELEASE PIPELINE

Even though the pipelines can be used to also release to UAT and PROD environments, the recommended method to release to these environments is to use the releases feature of DevOps. The steps below demonstrate how to configure the release pipeline so that release managers can issue releases based on builds that were ran by the build integration pipeline (CI).

Azure DevOps cloudstrucc / PowerPlatform-CIDC / Pipelines / Releases

PowerPlatform-CIDC

Overview Boards Repos Pipelines Environments Releases Library Task groups

Search all pipelines

+ New

New release pipeline Import release pipeline

**New release pipeline**

Releases Deployments

Releases

Release-57 \_2022061

Release-56 \_2022061

Release-55 \_2022061

Release-54 \_2022061

Release-53 \_2022061

Select the PowerPlatform-Deployment JSON file provided with this extension and press "OK"

**Import release pipeline**

Upload a release pipeline JSON file to import its contents

PowerPlatform-Deployment.json

22.3 KB remove

Cancel OK

Once imported you will be asked to specify your agent pool. You can select “Azure Pipelines” or your own custom pool and “windows-latest” as the agent specifications (for both UAT and PROD)

All pipelines > New release pipeline - Copy

PipelineTasksVariablesRetentionOptionsHistory

Artifacts | + Add

Stages | + Add

ARTEFACTS

Schedule not set

UAT Deployment ...

1 job, 11 tasks

Production Deplo...

1 job, 1 task

UAT Deployment Stage

Deployment process

Agent job

Run on agent

PowerShell Script

Power Platform Tool Installer

Power DevOps Tool Installer

Import Configuration Migration Data

Power Platform Import Solution

Import Configuration Migration Data 2

Power Platform Import Solution 2

Import Configuration Migration Data 3

Power Platform Import Solution 3

Import Configuration Migration Data 4

Power Platform Import Solution 4

Agent job

Remove

Display name \*

Agent job

Agent selection

Agent pool | Pool information | Manage

Azure Pipelines

Agent Specification \*

windows latest

Demands

Name	Condition	Value
powershell	exists	

+ Add

Execution plan

Parallelism

None

Multi-configuration

Multi-agent

Timeout \*

0

Job cancel timeout \*

Once specified, return to the release pipeline and the errors should be gone. You’re release pipeline is now ready for use.

