



**365 POWERUP
PHILLY!**

January 25, 2020

**Microsoft Technology
Center**

Malvern, PA

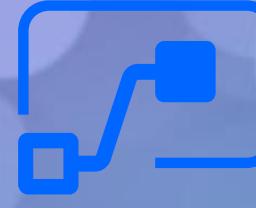


Microsoft

Solgari

eXalents

365 POWERUP



DevOps for The Small Implementation

Nick Doelman

Philadelphia, 2020

Nick Doelman

Power Platform/Dynamics 365 Specialist

Microsoft MVP - Business Applications

Microsoft Certified Trainer (MCT)

Ottawa, ON, CANADA

EMAIL: nick.doelman@readybms.com

TWITTER: @readyxrm

<https://readyxrm.blog>

<https://www.linkedin.com/in/nickdoelman/>

<https://github.com/readyxrm/simplealm>



What is DevOps?

Academics and practitioners have not developed a unique definition for the term "DevOps."

Len Bass, Ingo Weber, and Liming Zhu — three computer science researchers from the CSIRO and the Software Engineering Institute:

"a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality".



What is ALM?

- *Application lifecycle management (ALM) is the product lifecycle management (governance, development, and maintenance) of computer programs. It encompasses requirements management, software architecture, computer programming, software testing, software maintenance, change management, continuous integration, project management, and release management.*



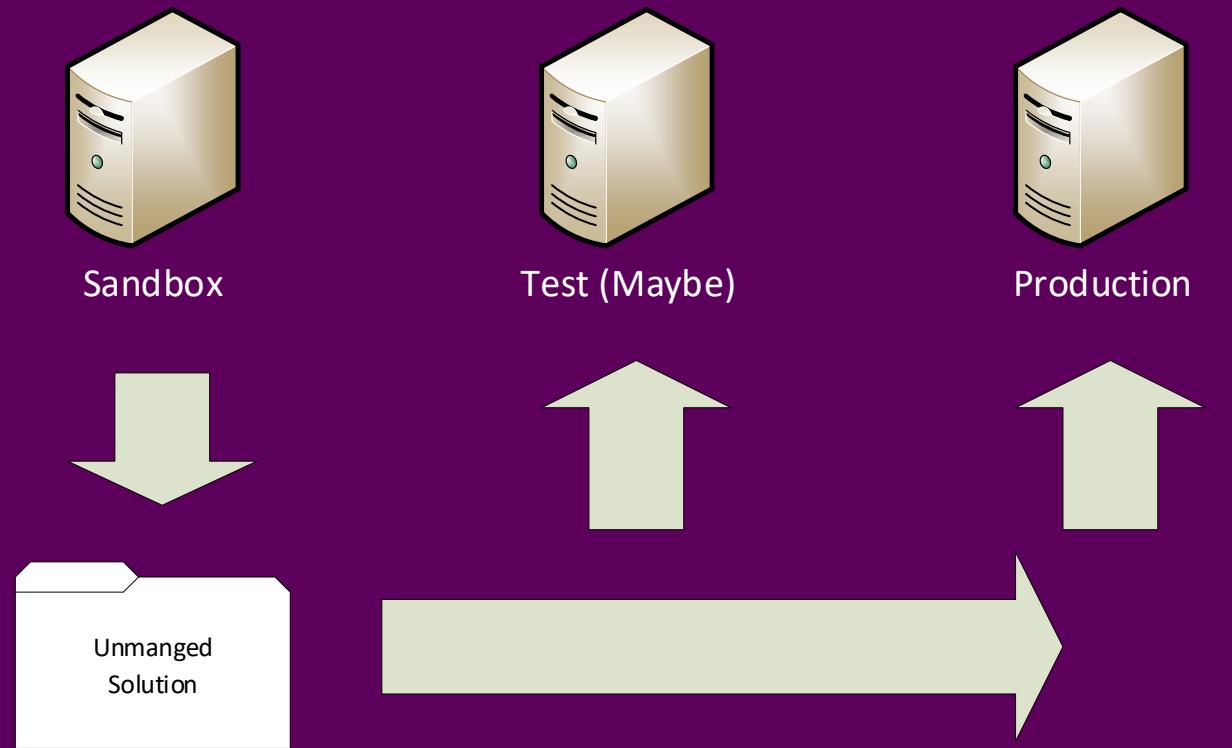
Why DevOps/ALM in small projects?

- A process to not mess up your Dynamics 365/Power Platform production environments.
- Ability to sleep at night knowing your work is safe.
- Being able to replicate the functionality of your environments.
- Allowing your work to not mess up someone else's work or vice versa.



Typical Solution Lifecycle in small Power Platform projects

- Solution is built or modified on one sandbox
- Unmanaged Solution is exported to a file
- Unmanaged solution is imported to Test (maybe) or imported to Prod
- No backups of solution (Sandbox is backed up)
- No way to see what might have been changed
- Updates may be done to Prod



Does this process seem familiar?

- Do you backup each version of your Dynamics 365/CDS Solution files somewhere?
- Do you have a copy of both the managed and unmanaged versions of your solutions?
- Do you know which copy was actually deployed to your system?
- Do you rely on a restore of a backup of your production environment every time you need a new sandbox environment?



Why the bad habits? – It's not your fault

- Export/Import buttons are what comes “out of the box”
- Assumption that Sandbox database is backed up, therefore all customizations are backed up
- Easy to “fix” unmanaged solutions in production
- Only until recently, typical deployments only had a sandbox and production instance
- Simple process, export -> import

All Solutions						
	Name	Display Name	Version	Installed On	Package T...	P
<input checked="" type="checkbox"/>	ContosoAssessments	Contoso Assessments	1.0.0.0	1/11/2020	Unmanag...	E
	ContosoDeviceOrdering	Contoso Device Order...	1.0.0.0	1/11/2020	Unmanag...	F
	Test	Test	1.0.0.1	1/11/2020	Unmanag...	F
	UpdatePortal	UpdatePortal	1.0.0.1	12/27/2019	Unmanag...	F
	CDSStarterPortal	Common Data Service...	9.2.10.10	11/25/2019	Managed	M



Issues

- Sandbox is a full database backup and only kept for 30 days
- Need to restore full database to “get back” what you needed, meaning you may need to overwrite your existing sandbox, losing current work
- Cannot remove unmanaged changes to an instance
- Potentially sandbox and prod can get out of sync
- Production gets cluttered with unmanaged changes
- More than one developer can overwrite other’s work
- No way to compare work
- Export/Import process tedious and prone to error



Restoring Prod as a Dev Instance

- Links to external systems (SharePoint, Exchange, Integrations, etc)
- Large database size
- User confusion
- Need to have unmanaged solutions
- Consider restoring Prod as a testing instance
- Unknown Microsoft patches, updates



Unmanaged vs Managed

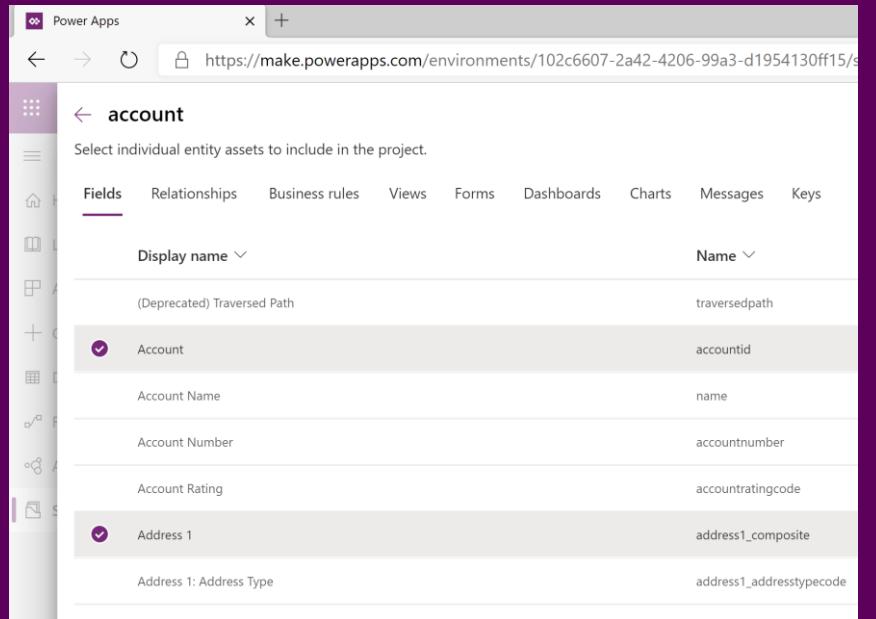


- Changes are made to unmanaged layer
- No ability to remove solutions, except manually piece by piece
- No organization or solution layering
- Microsoft direction of solution management -> may become mandated
- Does require good ALM practices
- Less issues now than early days (CRM 2011 – 2013)



Solution Health Check

- ✓ Solution segmentation?
- ✓ Supported techniques only?
- ✓ Single publisher?
- ✓ Solution Checker?
- ✓ Separate environments for each solution?
- ✓ Only have your managed dependencies installed?



ALM Health Check

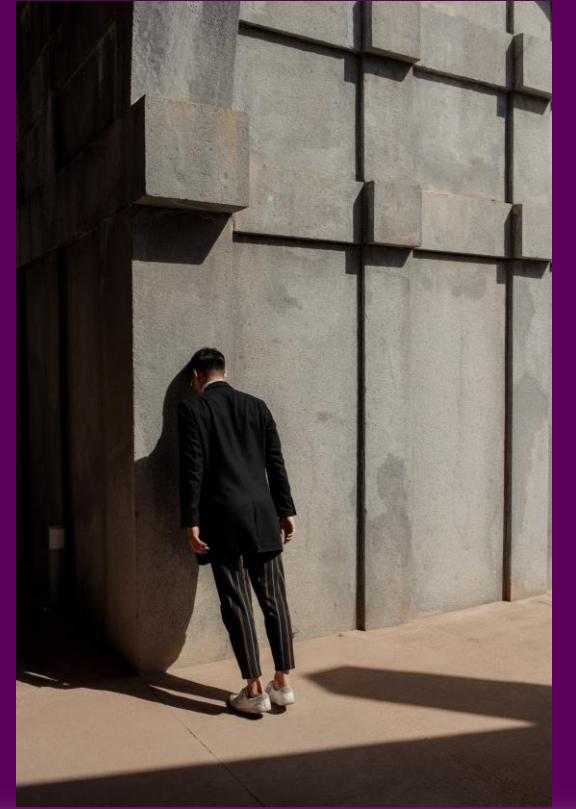
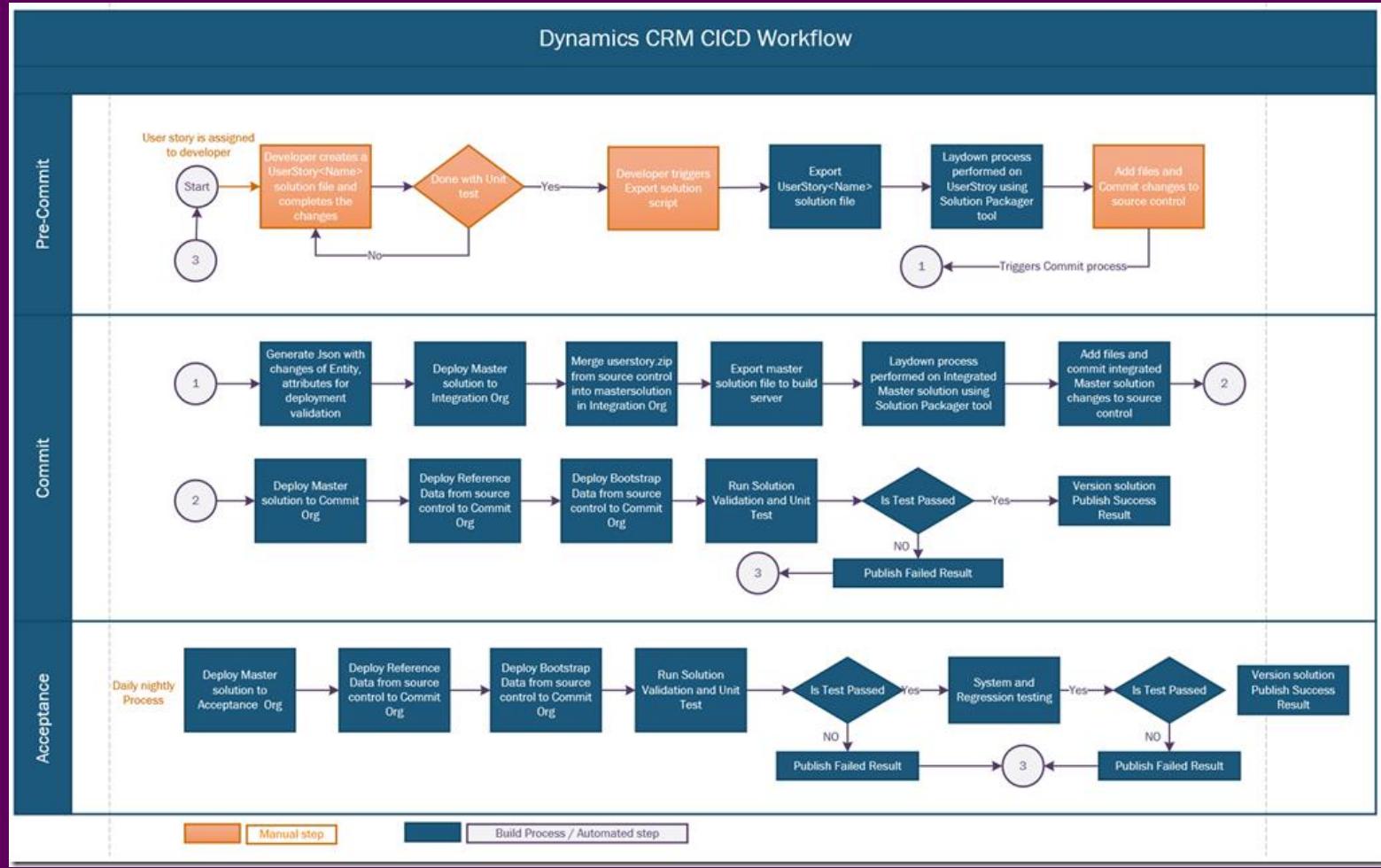
0

- ✓ Are you shipping as managed?
- ✓ Are your dev environments disposable?
- ✓ Source control definitive source of truth
- ✓ Using Solution Packager?
- ✓ Service production while working on v-next?
- ✓ 100% automated ALM?
- ✓ Developer isolation?

100



DevOps/CI to the Rescue?



365 POWERUP

Best Practices for Small Projects

- Start Simple!
- Focus on Building Apps
- One dev instance per App
- Development/Configuration in source control!
- Simple automation of solutions, will save time!
- Yes, managed solutions to production!



Easy Process to Export, Source Control and Import Solutions

Goal:

- With one command or one-click:
 - Export a solution from DEV
 - Unpack the solution
 - Commit to Source Control
 - Import a Managed version to TEST or PROD



Tools

- PowerShell
- Azure Dev Ops
- Solution Packager
- Git



What is PowerShell?

- PowerShell is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and associated scripting language.
- Used for many “automation” type work.
- Many of Microsoft tools are PowerShell enabled for automation (e.g. Configuration Migration Tool)

```
param (
[Parameter(Mandatory=$True)][ValidateNotNull()][string]$solutionShortName,
[string]$username =
[string]$password =
[string]$exportLocation = "C:\D365DEV\ReadyBMS\projectapp",
[string]$instance = "https://org2cb27713.crm3.dynamics.com",
[string]$region = "CAN",
[string]$type = "Office365",
[string]$organisation = "org50935876"
)

Import-Module Microsoft.Xrm.Data.PowerShell

$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credentials = New-Object System.Management.Automation.PSCredential ($username, $securePas

$CRMConn = Get-CrmConnection -Credential $credentials -DeploymentRegion $region -OnlineTyp

Write-Host "Exporting Solution, please wait."

Export-CrmSolution $solutionShortName $exportLocation -SolutionZipFileName $solutionShortN

Export-CrmSolution $solutionShortName $exportLocation -SolutionZipFileName $solutionShortN

Write-Host "Expanding Solution, please wait."

Expand-CrmSolution -ZipFile $solutionShortName"_Unmanaged.zip" -PackageType Unmanaged -Fo

Write-Host "Staging Changes to Source Control"

git add -A

Write-Host "Committing Changes to Source Control"

$today = Get-Date -Format g

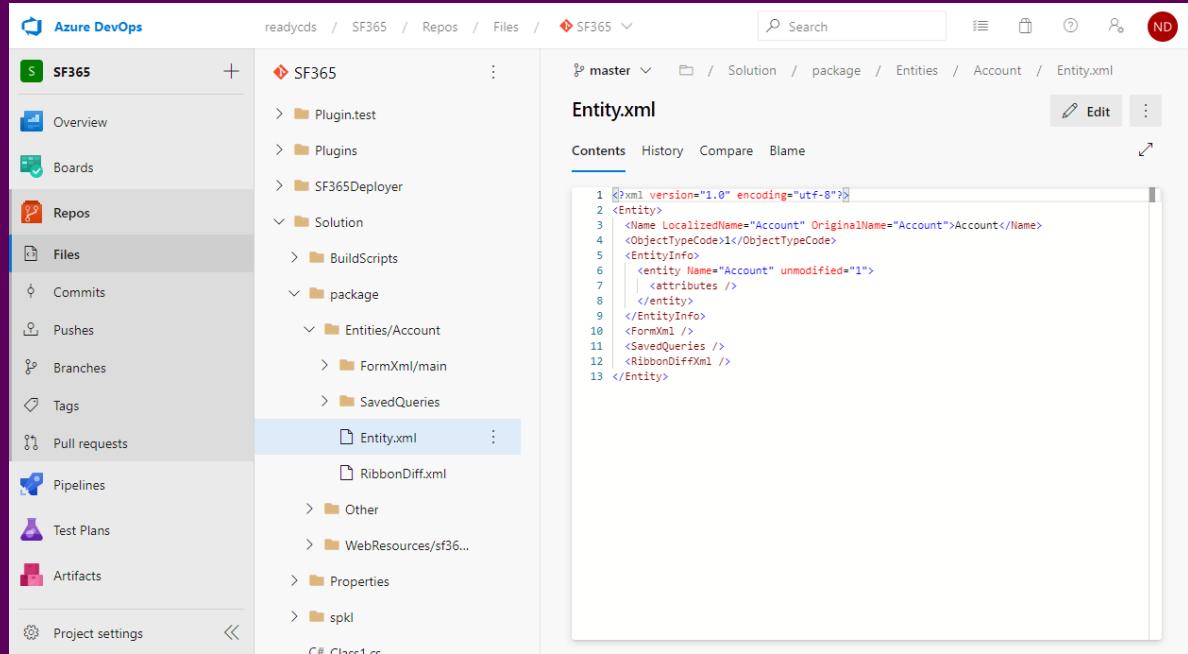
git commit -m "D365 Solution $($today)"

Write-Host "Syncing changes to Source Control"

git push|
```



What is Azure Dev Ops?

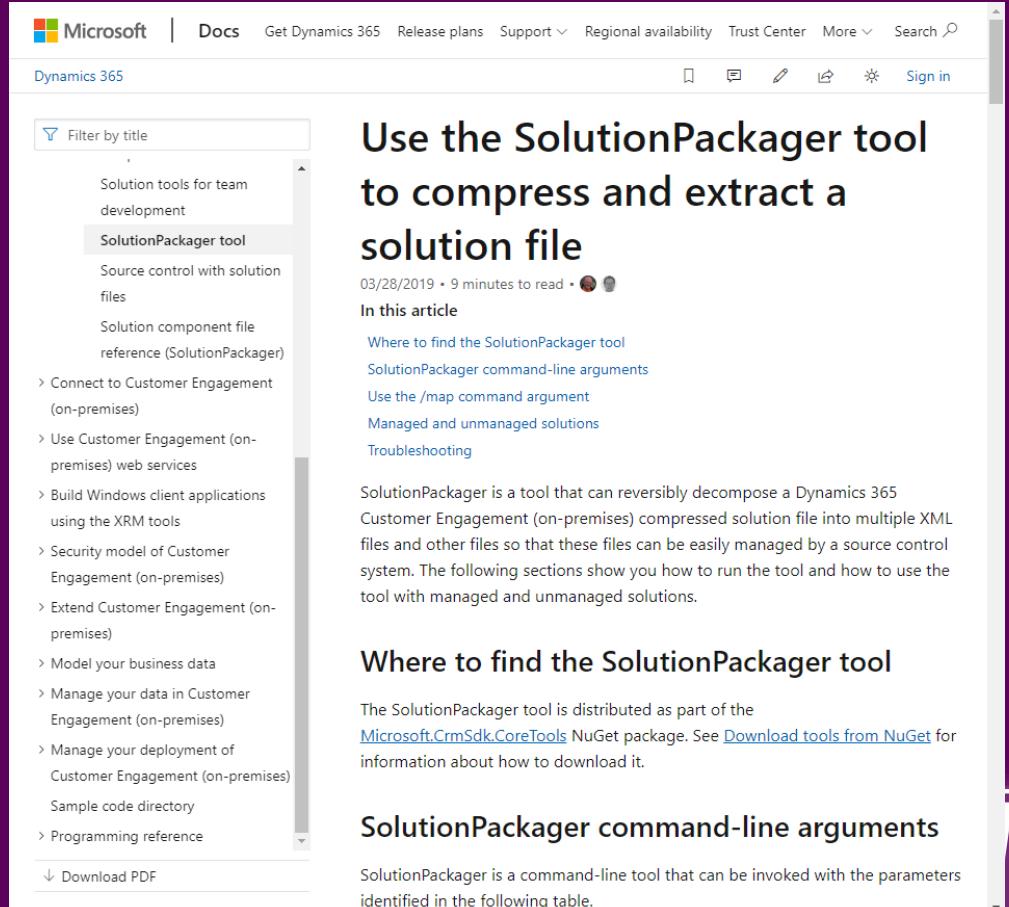


- Evolved from Team Foundation Services
- Cloud based source control
- Development Task Management
- Automated Testing Platform
- ALM/CI Tool (more on this later)



What is Solution Packager

- Command line tool (PowerShell)
- Extracts and Compresses (Packs) Dynamics 365/CDS solution files
- Why? Store extracted solution files in source control for viewing and comparison
- Ability to repack merged XML (multi-dev) and other code (plug-ins, etc.)



The screenshot shows a Microsoft Dynamics 365 documentation page. The top navigation bar includes links for Microsoft Docs, Get Dynamics 365, Release plans, Support, Regional availability, Trust Center, More, and Sign in. A search bar is also present. The main content area has a title "Use the SolutionPackager tool to compress and extract a solution file" with a date of 03/28/2019 and a reading time of 9 minutes. Below the title is a section titled "In this article" with links to "Where to find the SolutionPackager tool", "SolutionPackager command-line arguments", "Use the /map command argument", "Managed and unmanaged solutions", and "Troubleshooting". The main text explains that the SolutionPackager tool decomposes a compressed solution file into XML files and other files for management. At the bottom of the page is a "Download PDF" link.

Where to find the SolutionPackager tool

The SolutionPackager tool is distributed as part of the [Microsoft.CrmSdk.CoreTools](#) NuGet package. See [Download tools from NuGet](#) for information about how to download it.

SolutionPackager command-line arguments

SolutionPackager is a command-line tool that can be invoked with the parameters identified in the following table.



What is Git?

- Open source de-centralized source control system
- Easy to use
- Command line driven
- Industry standard (Used in Azure Dev Ops)



PowerShell Method

DEMO



High Level Steps

- Available on <https://readyxrm.blog/2019/10/31/simple-alm-for-dynamics-365-cds-projects/>
- Install Git
- Setup a Project in Azure Dev Ops
- Setup Local Repository (local files)
- Install Microsoft.Xrm.Data.PowerShell
- Install Adoxio.Dynamics.DevOps
- Install Power Platform Developer Tools



Build PowerShell Script to Export

- Accept a solution name as a parameter (or prompt you for one)
- Connect to your Dynamics 365/CDS Instance
- Export your solution, both as Managed and UnManaged
- Unpack the unmanaged solution
- Add, Commit and Push the Solution to source control (Azure DevOps) using Git commands.



C: > D365DEV > ReadyBMS > projectapp > [crmexportdemo.ps1](#) > ...

```
1 param (
2     [Parameter(Mandatory=$True)][ValidateNotNull()][string]$solutionShortName,
3     [string]$username = "nick.doelman@readybms.com",
4     [string]$password = "xxxx",
5     [string]$exportLocation = "C:\D365DEV\ReadyBMS\projectapp",
6     [string]$instance = "https://org2cb27713.crm3.dynamics.com",
7     [string]$region = "CAN",
8     [string]$type = "Office365",
9     [string]$organisation = "org50935876"
10    )
11
12 Import-Module Microsoft.Xrm.Data.PowerShell
13
14 $securePassword = ConvertTo-SecureString $password -AsPlainText -Force
15 $credentials = New-Object System.Management.Automation.PSCredential ($username, $securePassword)
16
17 $CRMConn = Get-CrmConnection -Credential $credentials -DeploymentRegion $region -OnlineType $type -Or
18
19 Write-Host "Exporting Solution, please wait."
20
21 Export-CrmSolution $solutionShortName $exportLocation -SolutionZipFileName $solutionShortName"_Unmanaged.zip"
22 Export-CrmSolution $solutionShortName $exportLocation -SolutionZipFileName $solutionShortName"_Managed.zip"
23
24 Write-Host "Expanding Solution, please wait."
25 Expand-CrmSolution -ZipFile $solutionShortName"_Unmanaged.zip" -PackageType Unmanaged -Folder $solutionShortName
26
27 Write-Host "Staging Changes to Source Control"
28 git add -A
29
30 Write-Host "Committing Changes to Source Control"
31 $today = Get-Date -Format g
32 git commit -m "D365 Solution $($today)"
33
34 Write-Host "Syncing changes to Source Control"
35 git push
36
```



Build PowerShell Script to Import

- Take the solution name as a parameter (assuming it was exported and still exists in the local repository)
- Connect to your destination Dynamics 365/CDS instance.
- Import the solution, if you import the unmanaged, it will publish as well.



```
C: > D365DEV > ReadyBMS > projectapp > ➜ crmimportdemo.ps1 > ...

1 param (
2     [Parameter(Mandatory=$True)][ValidateNotNull()][string]$solutionShortName,
3     [string]$username = "nick.doelman@readybms.com",
4     [string]$password = "xxxx",
5     [string]$solutionPath = "C:\D365DEV\ReadyBMS\projectapp",
6     [string]$instance = "https://readybms.crm3.dynamics.com",
7     [string]$region = "CAN",
8     [string]$type = "Office365",
9     [string]$organisation = "readybms"
10    )
11
12
13 Import-Module Microsoft.Xrm.Data.PowerShell
14
15
16 $securePassword = ConvertTo-SecureString $password -AsPlainText -Force
17 $credentials = New-Object System.Management.Automation.PSCredential ($username, $securePassword)
18
19 $CRMConn = Get-CrmConnection -ServerUrl $instance -Credential $credentials -OrganizationName $organisation
20
21 Set-CrmConnectionTimeout -TimeoutInSeconds 3600 -conn $CRMConn
22
23 Write-Host "Importing Solution, please wait."
24
25 Write-Host "$($solutionPath)\$($solutionShortName)_Managed.zip"
26
27 Import-CrmSolution -SolutionFilePath "$($solutionPath)\$($solutionShortName)_Managed.zip" -conn $CRMConn
28
29
```





365 POWERUP

What are Power Apps Build Tools

The screenshot shows the Microsoft Azure DevOps Marketplace page for the "PowerApps BuildTools (0.1.10)" extension. The page includes the following details:

- Icon:** A purple square icon featuring a white PowerApps logo.
- Name:** PowerApps BuildTools (0.1.10)
- Provider:** Microsoft
- Installs:** 2,928 installs
- Rating:** ★★★★☆ (6)
- Description:** Automate common build and deployment tasks related to PowerApps
- Get it free button:** A green button at the bottom left.

Below the main card, there's a navigation bar with links for Overview, Q & A, and Rating & Review. The Overview section contains the following content:

PowerApps Build Tools for Azure DevOps

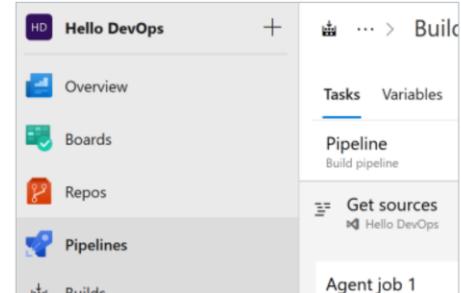
Overview

Use PowerApps Build Tools to automate common build and deployment tasks related to PowerApps. This includes synchronization of solution metadata (a.k.a. solutions) between development environments and source control, generating build artifacts, deploying to downstream environments, provisioning/de-provisioning of environments, and the ability to perform static analysis checks against your solution using the PowerApps checker service.

PRE-RELEASE SOFTWARE. The software is a pre-release version. It may not work the way a final version of the software will. We may change it for the final, commercial version. We also may not release a commercial version.

Table of Contents:

- What are PowerApps Build tools



Dynamics 365 Build Tools

The screenshot shows the Dynamics 365 Build Tools page in the Visual Studio Marketplace. At the top, there's a navigation bar with 'Visual Studio | Marketplace' and a search bar. Below it, the breadcrumb navigation shows 'Azure DevOps > Azure Pipelines > Dynamics 365 Build Tools'. The main content area features a large icon of a blue triangle with green arrows forming a circle. The title 'Dynamics 365 Build Tools' is displayed, along with the author 'Wael Hamze' and a rating of 4.5 stars from 45 reviews. A button labeled 'Get it free' is present. Below this, there's a section titled 'Dynamics 365 Build Tools' with a brief description: 'Dynamics 365 Build Tools is a set of tools that makes it easy and quick to automate builds and deployment of your PowerApps/CDS/Dynamics 365 CE solutions.' There are also sections for 'Compatibility' (listing Dynamics 365 versions 8.x.x and 9.x.x, and PowerApps/CDS), 'Build Definitions / Third Party Tools', and 'Categories' (listing Azure Pipelines). A note at the bottom states 'Works with Hosted VSTS Agents'.

VisualStudio | Marketplace

Azure DevOps > Azure Pipelines > Dynamics 365 Build Tools

 Dynamics 365 Build Tools

Wael Hamze | 6,427 installs | ★★★★☆ (45) | Preview

Tasks for automating Build & Deployment of Dynamics 365 CE/CDS/PowerApps Solutions

Get it free

Overview Q & A Rating & Review

Dynamics 365 Build Tools

Dynamics 365 Build Tools is a set of tools that makes it easy and quick to automate builds and deployment of your PowerApps/CDS/Dynamics 365 CE solutions.

This will allow you to setup a fully automated DevOps pipeline so you can deliver CRM more frequently in a consistent and reliable way.

Compatibility

Dynamics 365 (8.x.x)
Dynamics 365 (9.x.x)/CDS/PowerApps
(Some tasks may work with previous version of Dynamics CRM)

Azure DevOps/Azure DevOps Server/TFS For support and installation [instructions](#)

Works with Hosted VSTS Agents

Build Definitions / Third Party Tools

Build Options Repository

Save Undo

Categories

Azure Pipelines

Tags



Power Apps Build Tools Method

DEMO



Building a Pipeline

The screenshot shows the Azure DevOps Pipelines interface for a project named "DeviceManagement". The pipeline is titled "DeviceManagement-CI". The "Tasks" tab is selected. The pipeline consists of the following steps:

- Get sources**: Checks out the "DeviceManagement" repository from the "master" branch.
- Agent job 1**: Run on agent.
 - PowerApps Tool Installer**: PREVIEW PowerApps Tool Installer
 - PowerApps Export Solution**: PREVIEW PowerApps Export Solution
 - PowerApps Unpack Solution**: PREVIEW PowerApps Unpack Solution
- Command Line Script**: Command line.
 - Script**:

```
echo commit all changes
git config user.email "nick@airlift2020.onmicrosoft.com"
git config user.name "Automatic Build"
git checkout master
git add -all
git commit -m "solution init"
echo push code to new repo
git -c http.extraheader="AUTHORIZATION: bearer $System.AccessToken" push origin master
```
 - Advanced**
 - Control Options**
 - Environment Variables**
 - Output Variables**



Finding Changes

The screenshot shows the Azure DevOps interface for a project named "DeviceManagement". The left sidebar is the navigation menu with options like Overview, Boards, Repos, Files, Pipelines, Test Plans, Artifacts, and Project settings. The "Files" section is currently selected. In the main area, a file named "{2be0a20f-9142-40c7-bb9f-b9573ecb17cf}.xml" is open in a "Compare" view. The view shows two versions of the XML file: "f8aab9d2 (previous)" and "298d6478 (head)". The code editor highlights differences in the XML structure, such as new rows and labels in the head version. The XML code is as follows:

```
<cell id="{e70ee273-0b3...<labels><label description=</labels><control id="ownerid"</cell><row><cell id="{b49e1b5d-5d2...<labels><label description=</labels><control id="contoso_</cell><row><cell id="{af2d399c-8fc...<labels><label description=</labels><control id="contoso_</cell><row><cell id="...<rows><section><sections></column><column width="66%"><sections><section showlabel="false" sh<labels>
```



PowerShell vs Power Apps Build Tools

PowerShell

- Easy
- Easier to troubleshoot
- Can incorporate data (Portals?)
- Can trigger from desktop
- Can use other source control (GitHub, GitLab, etc)
- Merging and branching

Build Tools

- Still in Preview
- Less scripting
- Still need to write some code (Git)
- Can be scheduled



Next Steps!

- Automate Solution Export and Import
- Add to source control
- Unpack Solutions
- Develop a modular, layered solution strategy
- Only add discreet components to solutions (not all)
- Take advantage of multiple environments
- Invest time in learning tools



Thank your Sponsors!





**365 POWER UP
PHILLY!**

Please thank
your sponsors!



Microsoft

Solgari

eXalents