365 POWER UP PHILLY!

January 25, 2020

Microsoft Technology Center

Malvern, PA

Microsoft Solgari eXalents

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

Microsoft MVP Most Valuable Professional

D365UG/CRMUG ALL STAR

Microsoft Certified Trainer — Microsoft MCT Trainer 2019 - 2020

365 POWER UP

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".

365 POWER UP

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.*

365 POWER UP

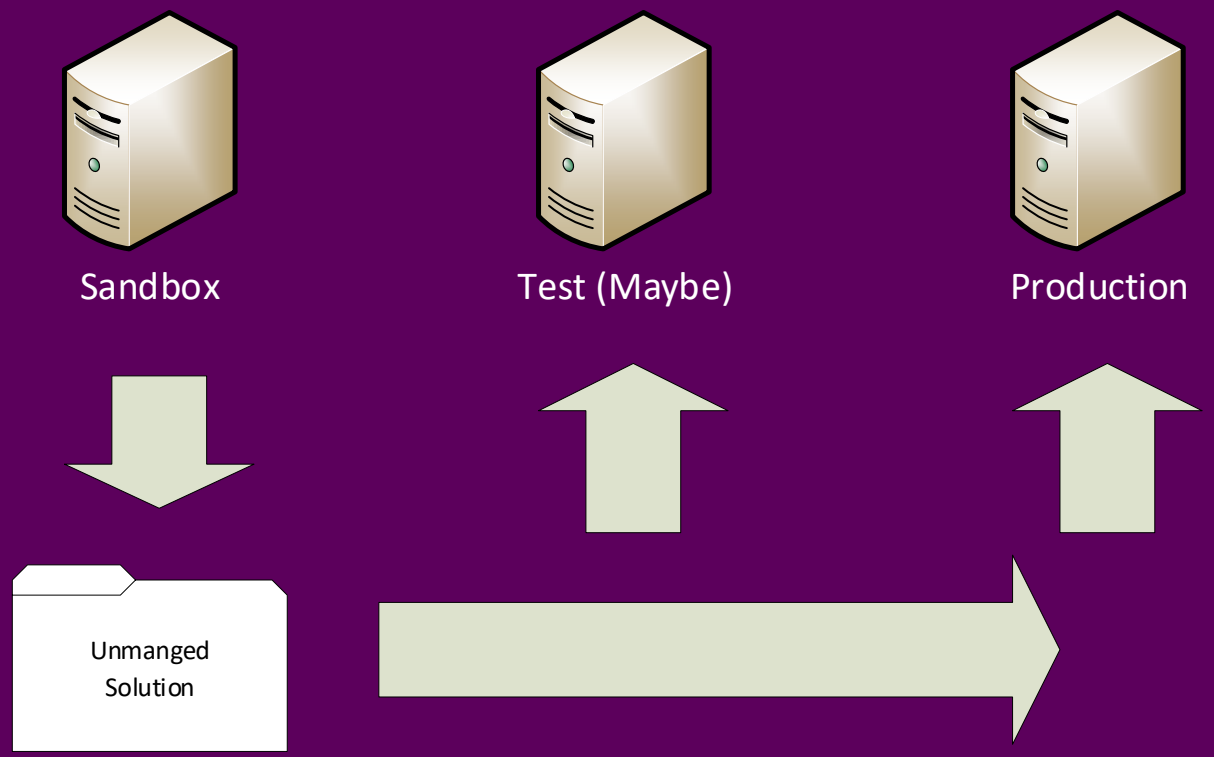
Why DevOps/ALM in small projects?

- 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.

365 POWER UP

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



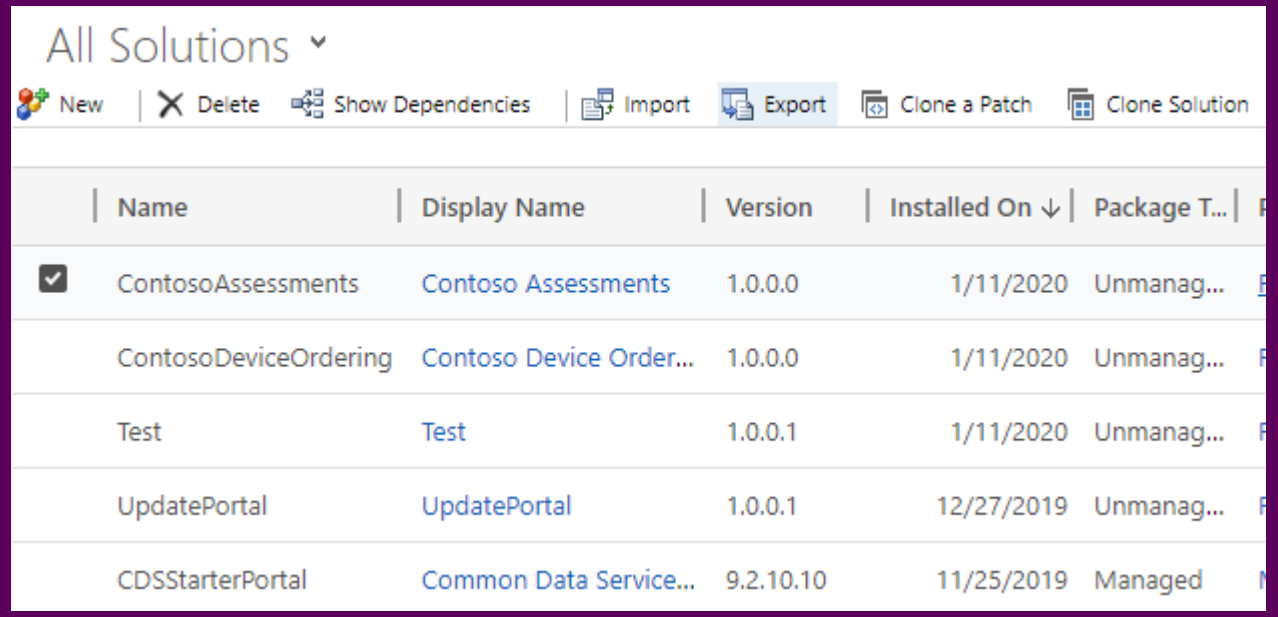
Does this process seem familiar?

- 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?

365 POWER UP

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

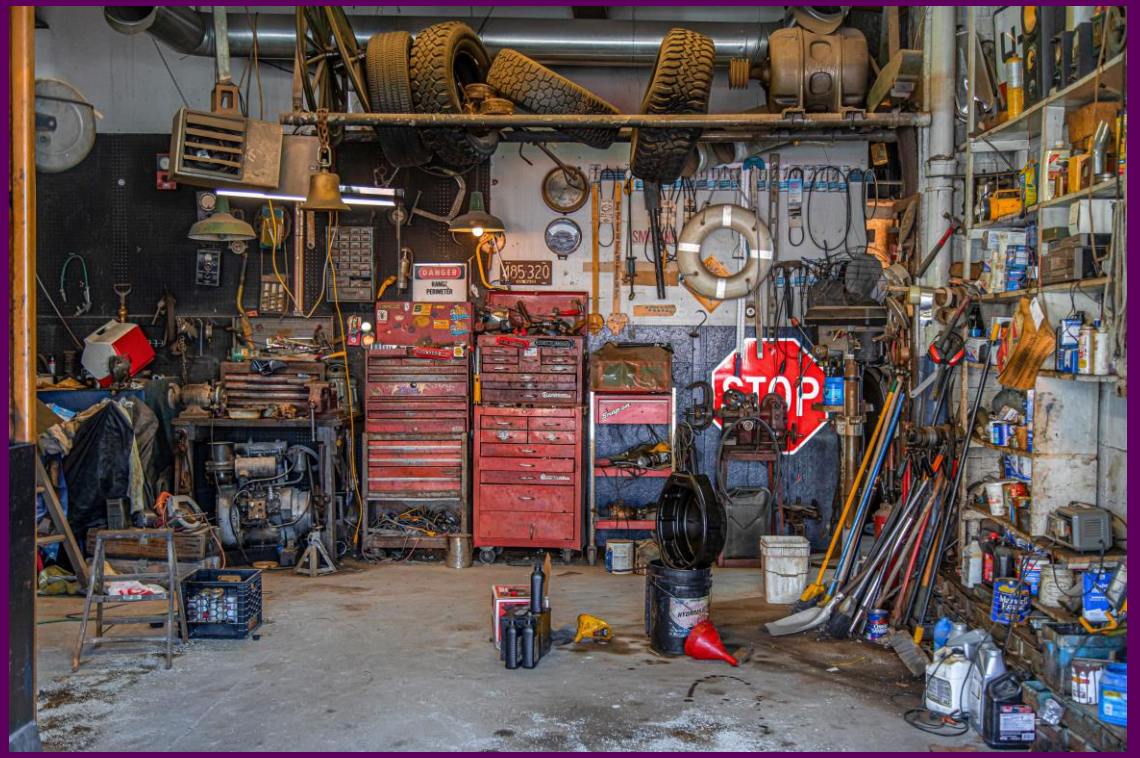
- Export/Import buttons are what comes "out of the box"
- Assumption that Sandbox database is backed up, therefore all customizations are backed up
- Easy to "fix" unmanaged solutions in production
- Only until recently, typical deployments only had a sandbox and production instance
- Simple process,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



365 POWER UP

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



365 POWER UP

Restoring Prod as a Dev Instance

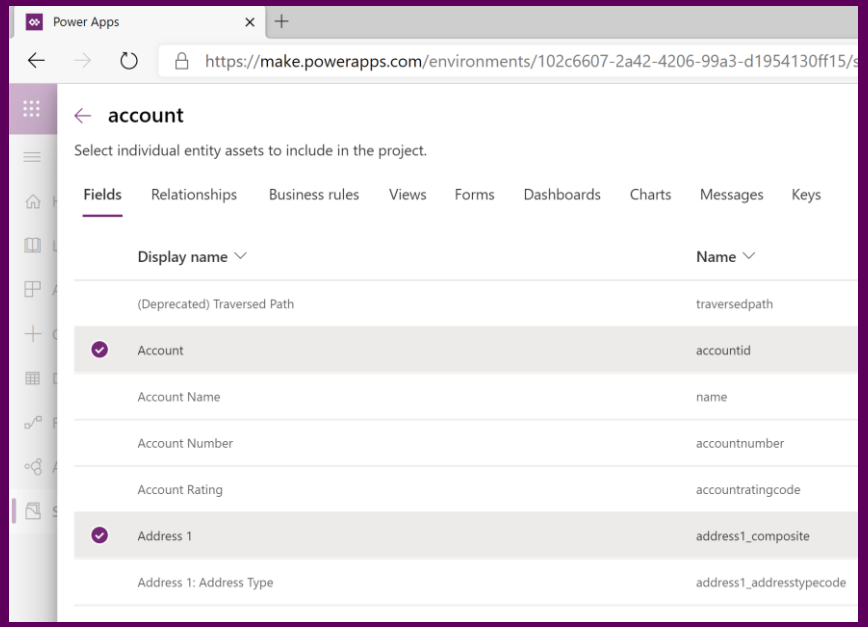
- Links to external systems (SharePoint, Exchange, Integrations, etc)
- Large database size
- User confusion
- Need to have unmanaged solutions
- Consider restoring Prod as a testing instance
- Unknown Microsoft patches, updates

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 then early days (CRM 2011 – 2013)

365 POWER UP

Solution Health Check



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

365 POWER UP

ALM Health Check

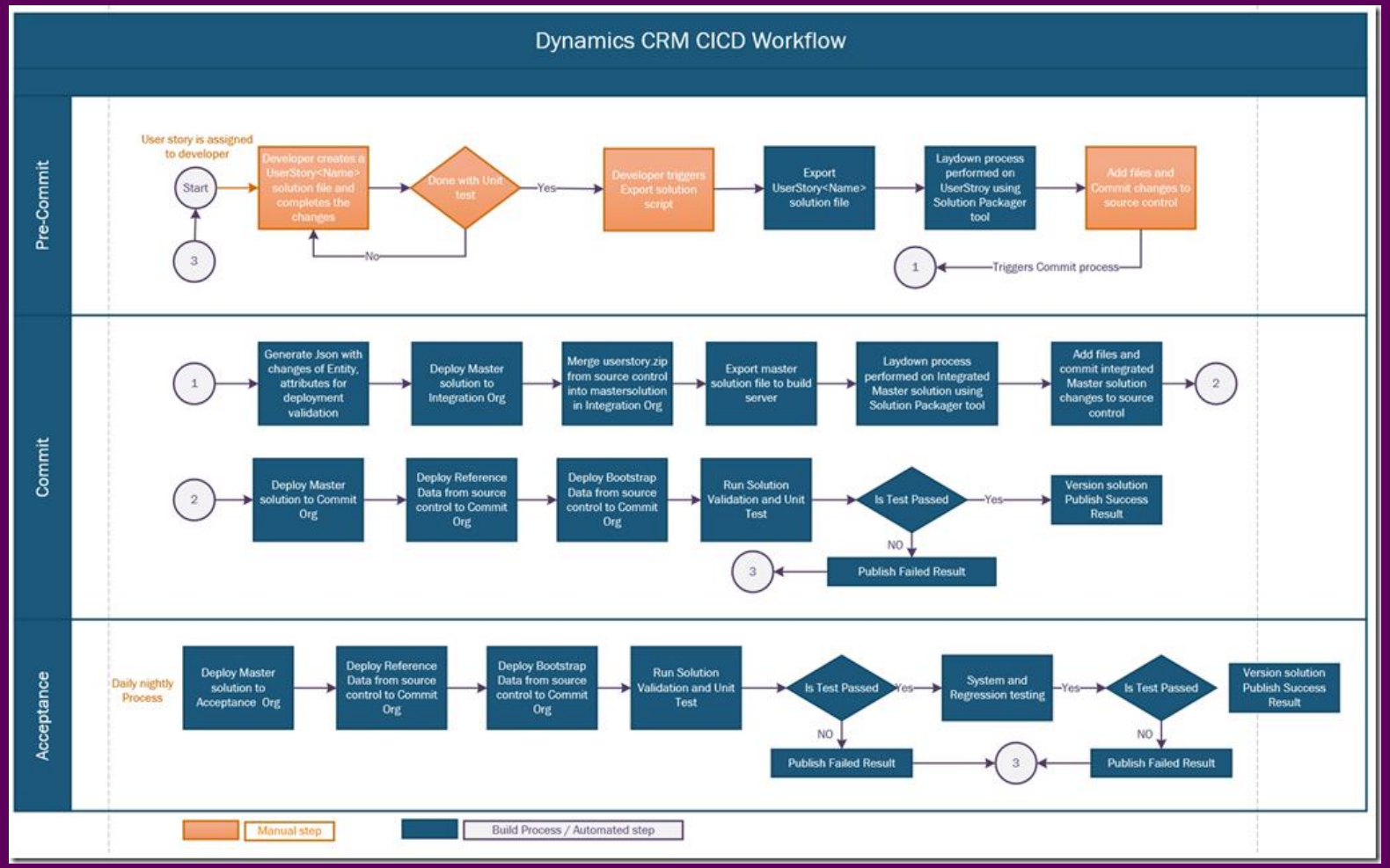
0

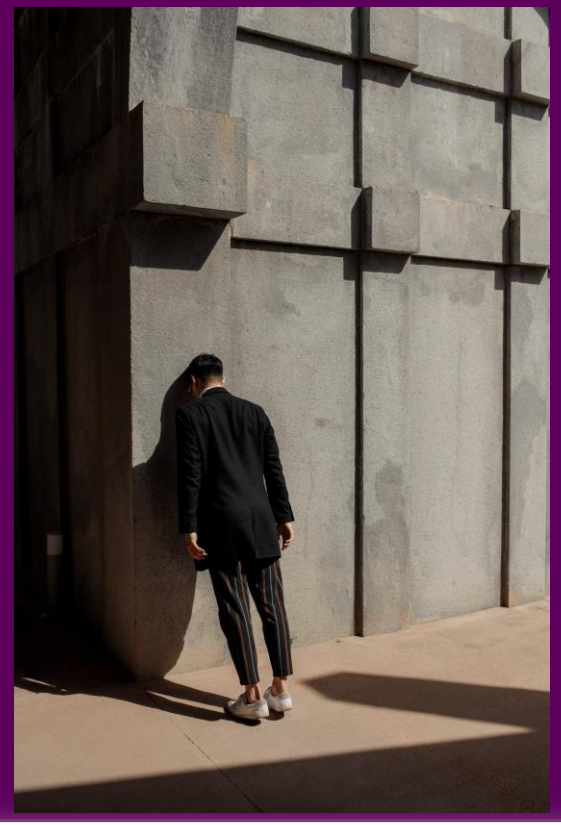
- ✓ Are you shipping as managed?
- ✓ Are your dev environments disposable?
- ✓ Source control definitive source of truth
- ✓ Using Solution Packager?
- ✓ Service production while working on v-next?
- ✓ 100% automated ALM?
- ✓ Developer isolation?

100

365 POWER UP

DevOps/CI to the Rescue?





365 POWER UP

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!

365 POWER UP

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

365 POWER UP

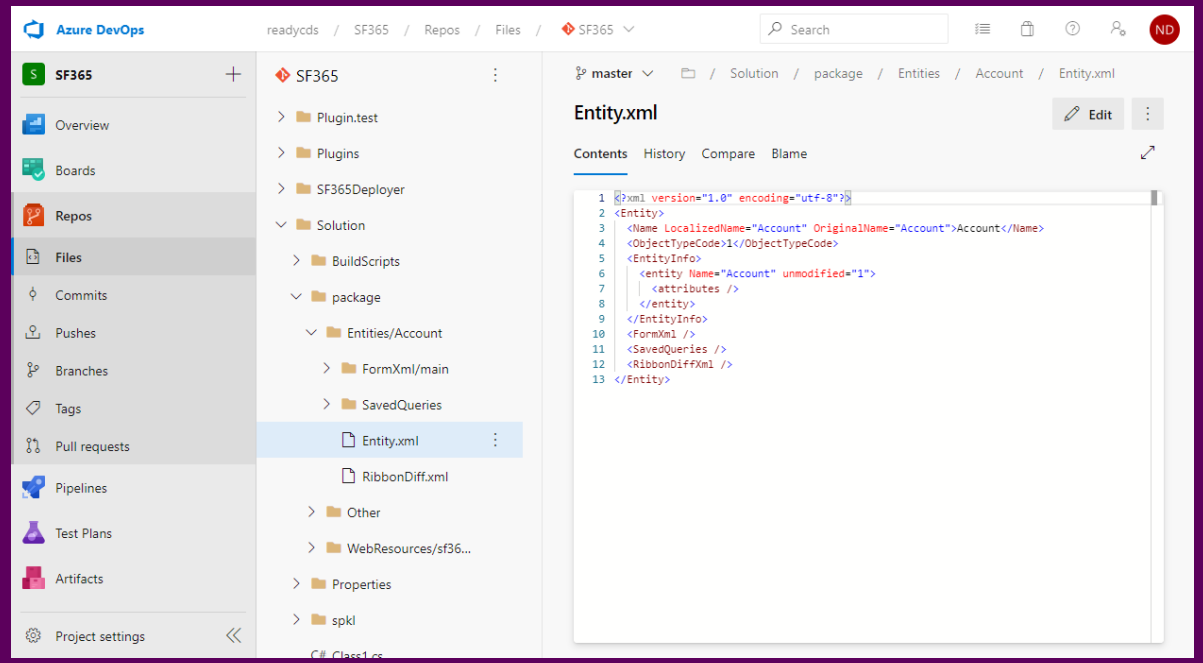
What is PowerShell?

- 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 -Fol

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
```

365 POWER UP

What is Azure Dev Ops?

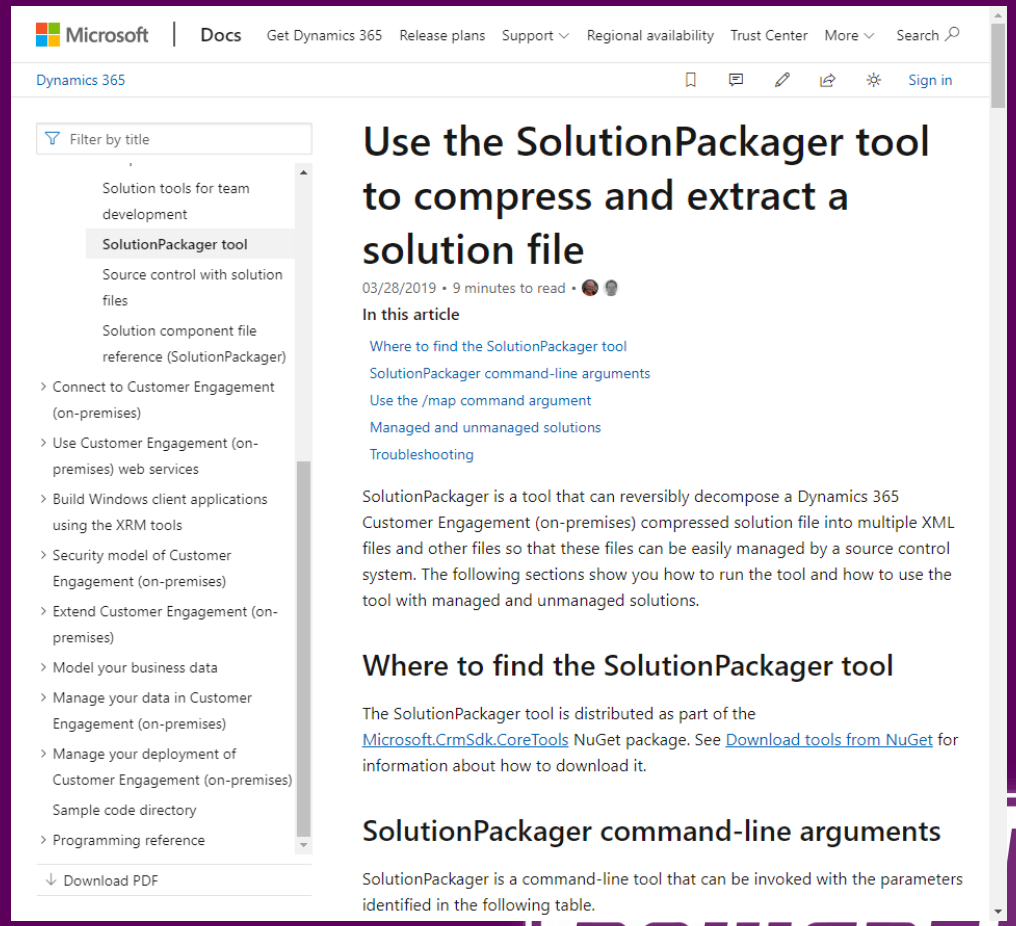


- Evolved from Team Foundation Services
- Cloud based source control
- Development Task Management
- Automated Testing Platform
- ALM/CI Tool (more on this later)

365 POWER UP

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.)



What is Git?

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

365 POWER UP

PowerShell Method

365 POWER UP

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 > ...

```powershell
param (
    [Parameter(Mandatory=$True)][ValidateNotNull()][string]$solutionShortName,
    [string]$username = "nick.doelman@readybms.com",
    [string]$password = "xxxx",
    [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, $securePassword)

$CRMConn = Get-CrmConnection -Credential $credentials -DeploymentRegion $region -OnlineType $type -Or

Write-Host "Exporting Solution, please wait."

Export-CrmSolution $solutionShortName $exportLocation -SolutionZipFileName $solutionShortName"_Unmana
Export-CrmSolution $solutionShortName $exportLocation -SolutionZipFileName $solutionShortName"_Manage

Write-Host "Expanding Solution, please wait."
Expand-CrmSolution -ZipFile $solutionShortName"_Unmanaged.zip" -PackageType Unmanaged -Folder $soluti

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

```

365 POWER UP

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 > ...

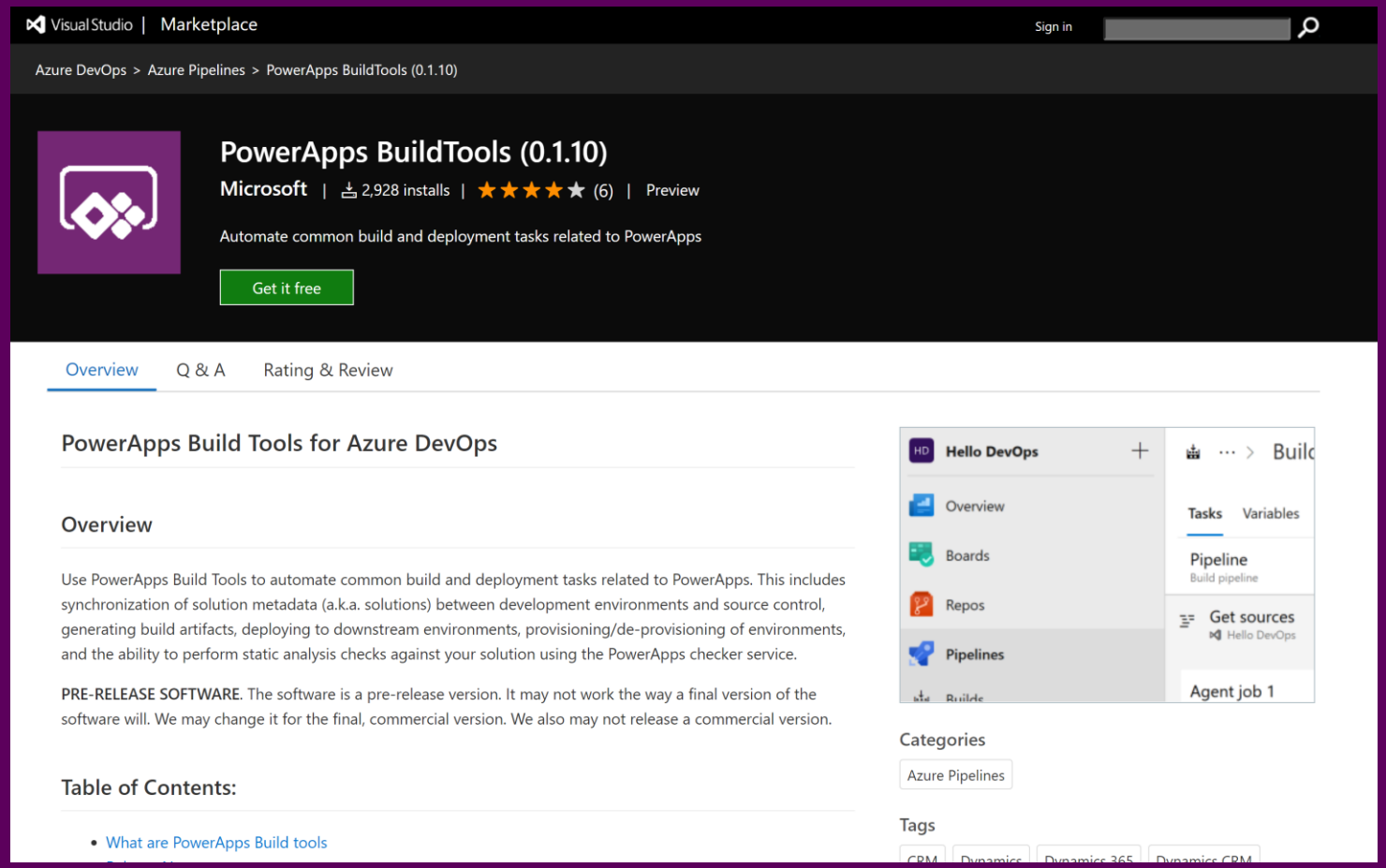
```powershell
param (
    [Parameter(Mandatory=$True)][ValidateNotNull()][string]$solutionShortName,
    [string]$username = "nick.doelman@readybms.com",
    [string]$password = "xxxx",
    [string]$solutionPath = "C:\D365DEV\ReadyBMS\projectapp",
    [string]$instance = "https://readybms.crm3.dynamics.com",
    [string]$region = "CAN",
    [string]$type = "Office365",
    [string]$organisation = "readybms"
    )


Import-Module Microsoft.Xrm.Data.Powershell


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

$CRMConn = Get-CrmConnection –ServerUrl $instance -Credential $credentials -OrganizationName $organisation

Set-CrmConnectionTimeout -TimeoutInSeconds 3600 -conn $CRMConn

Write-Host "Importing Solution, please wait."

Write-Host "$($solutionPath)\$($solutionShortName)_Managed.zip"

Import-CrmSolution -SolutionFilePath "$($solutionPath)\$($solutionShortName)_Managed.zip" -conn $CRMConn


```

365 POWER UP

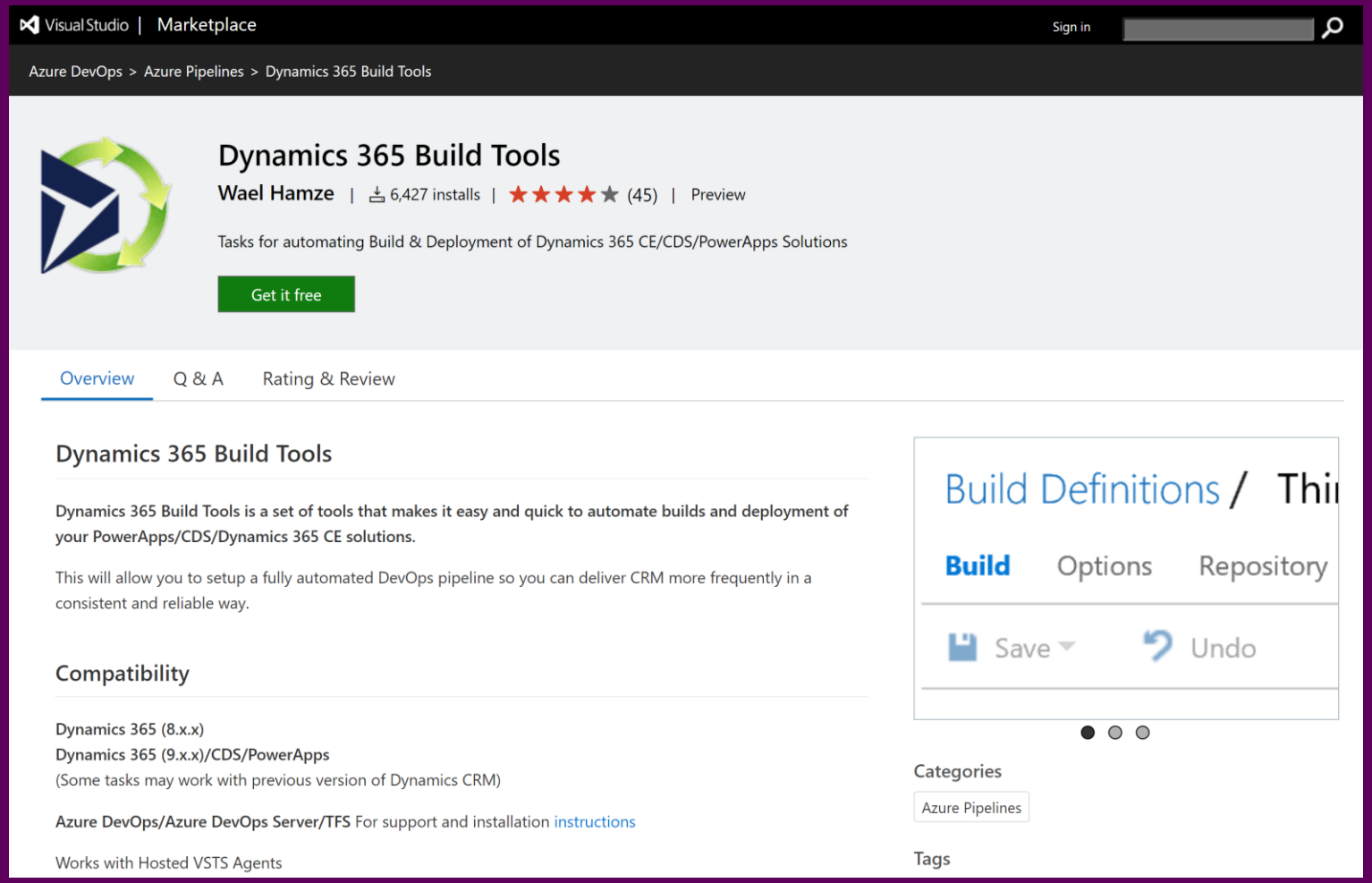

Nice blog post, why don't you use Power Apps Build Tools?
REDBLACKS

365 POWER UP

What are Power Apps Build Tools



Dynamics 365 Build Tools

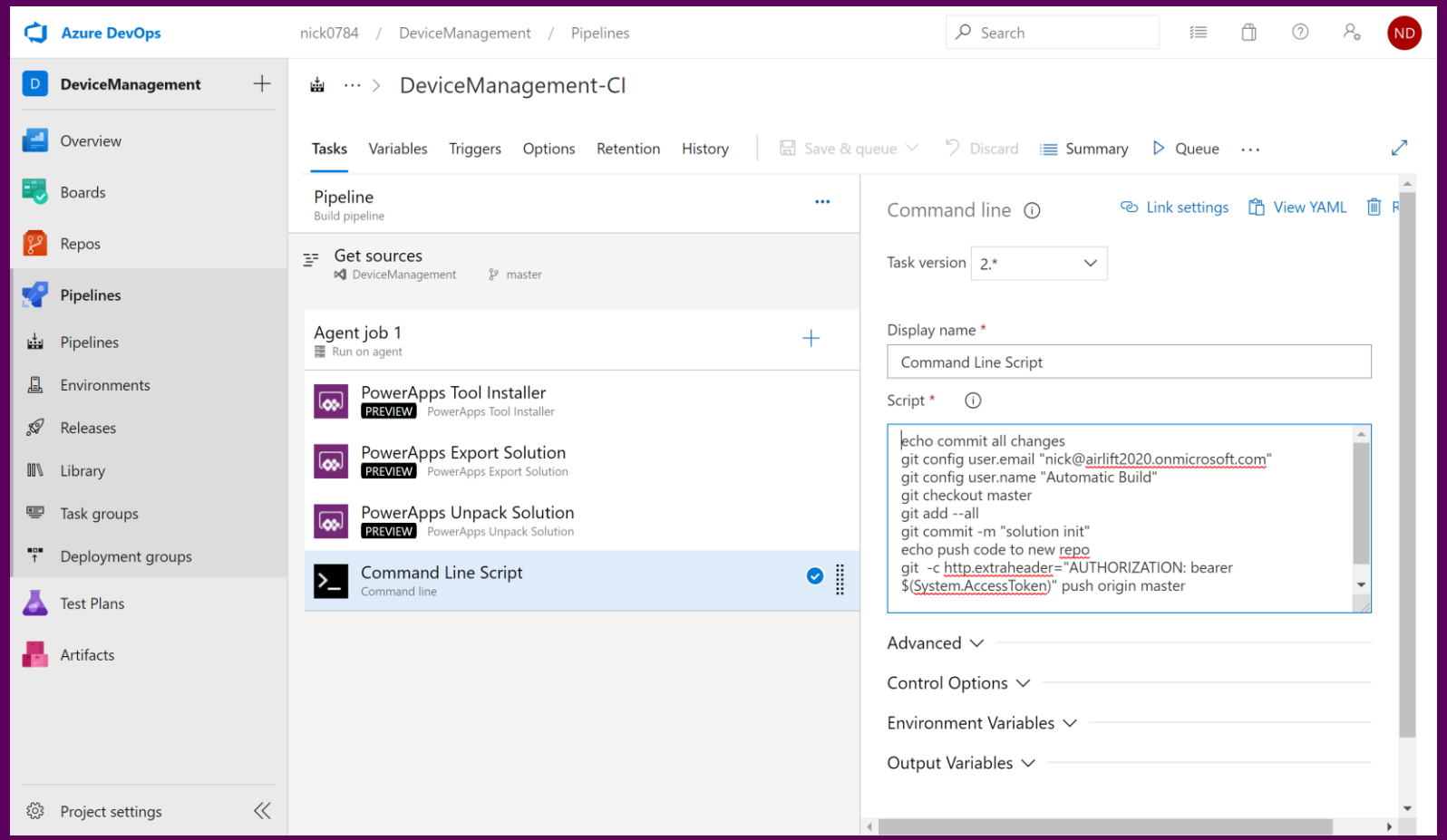


365 POWER UP

Power Apps Build Tools Method

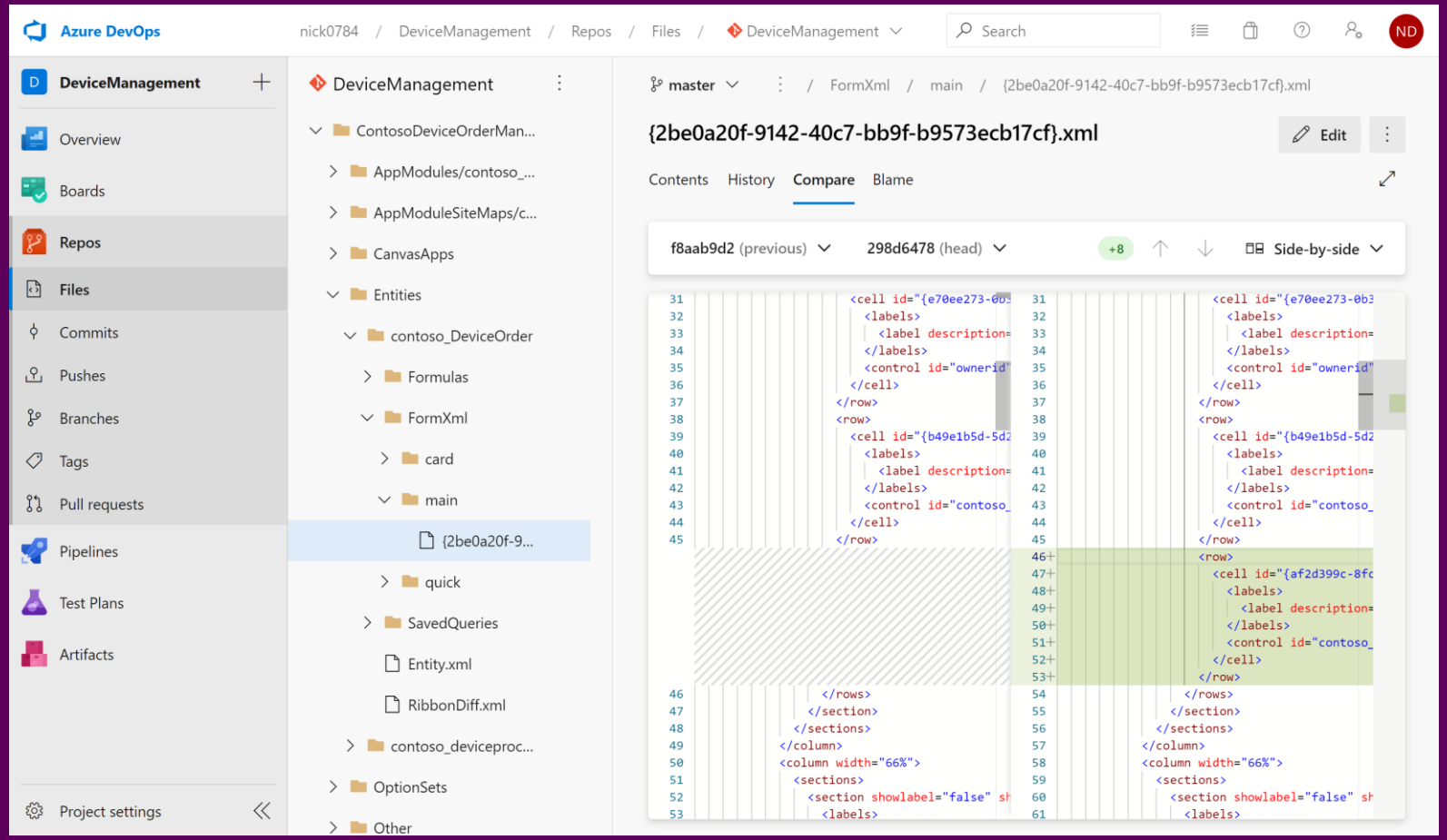
365 POWER UP

Building a Pipeline



365 POWER UP

Finding Changes



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

365 POWER UP

Thank your Sponsors!

365 POWER UP PHILLY!

Please thank your sponsors!

Microsoft

Solgari

eXalents