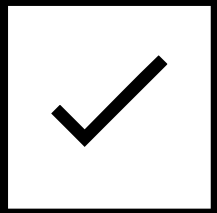
AUTODESK UNIVERSITY

Deploy Better Plug-ins Faster Through CI/CD and Unit Testing in Azure DevOps

SD501013

Andrea Tassera
Global Specialist Design Technology - Woods Bagot
@dre_tas

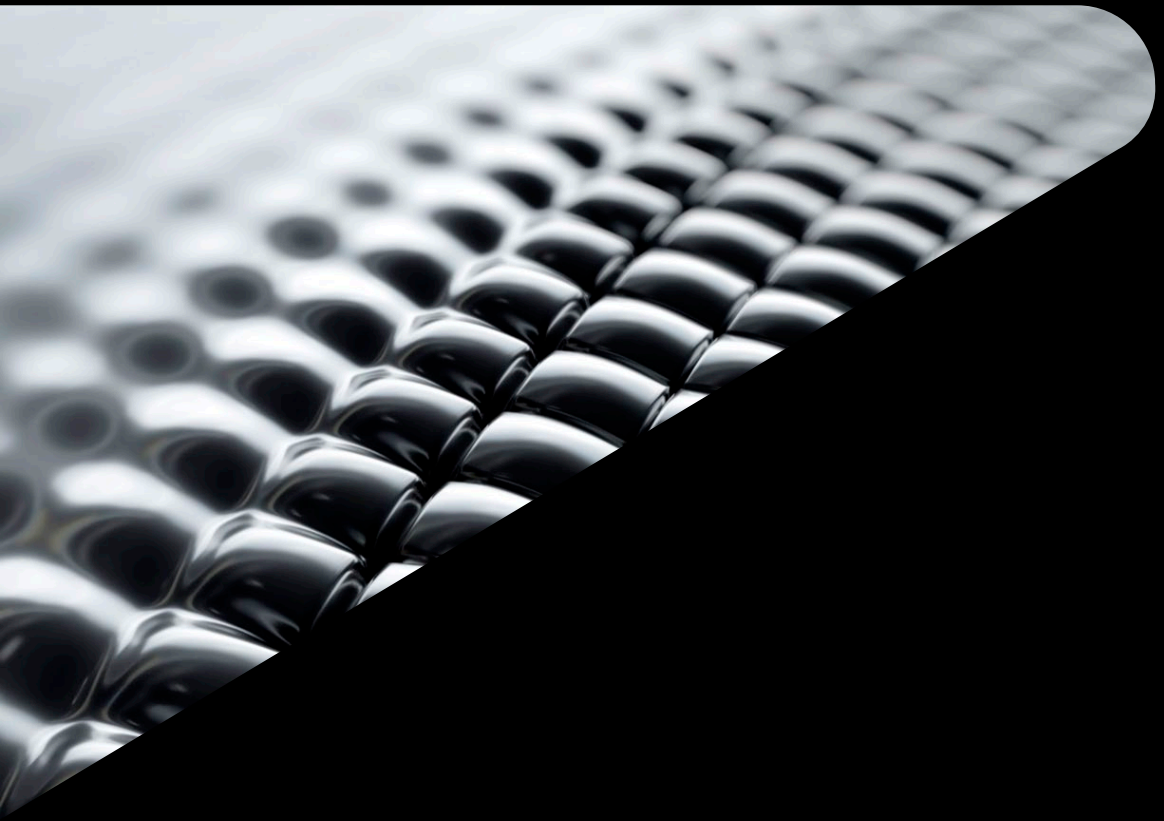
How long does it take you to deploy tools? 

How do you ensure quality? 

Why are you here

- **How long** does it take you **to release** a new tool?
  - Send files to users and tell them where to copy them;
  - Keep it to yourself and be the only one to use it for everyone;
  - Copy files in a server location;
  - Create package on a deployment system (e.g. Microsoft SCCM).

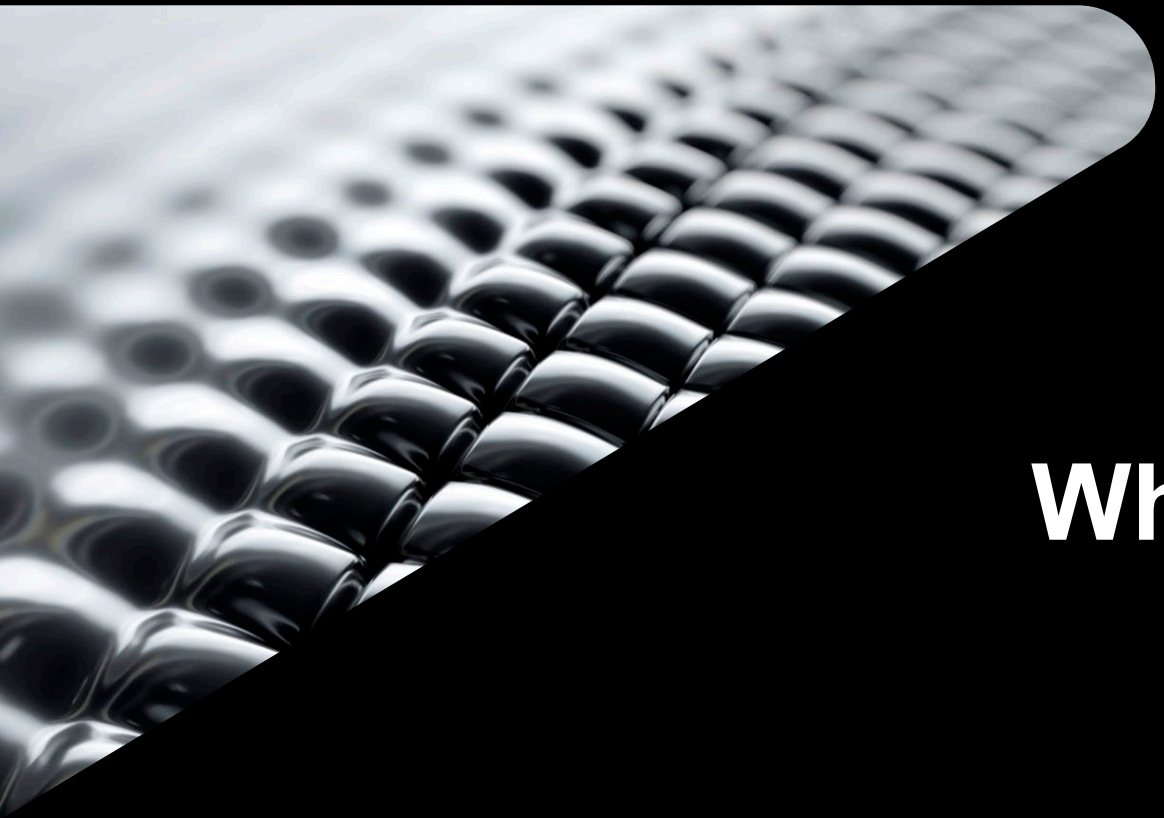
- How do you ensure **quality**?
  - Get experienced Revit guy to test;
  - Get mixed group of people to kick the tires;
  - Go rogue without testing.



Assumptions

A few assumptions

- Know from the beginning, this is “made by architects for architects”
- For that reason, we simplify things, and it isn’t entirely orthodox…but it works!
- We’ll focus on DevOps for Revit plugins (no web apps, etc)
- We work on a Microsoft environment, as often happens in design environments
- We work with a Visual Studio solution and NuGet packages
- This workflow refers specifically to pyRevit, but it can be adapted to any structure (we do something similar with Rhino as well)

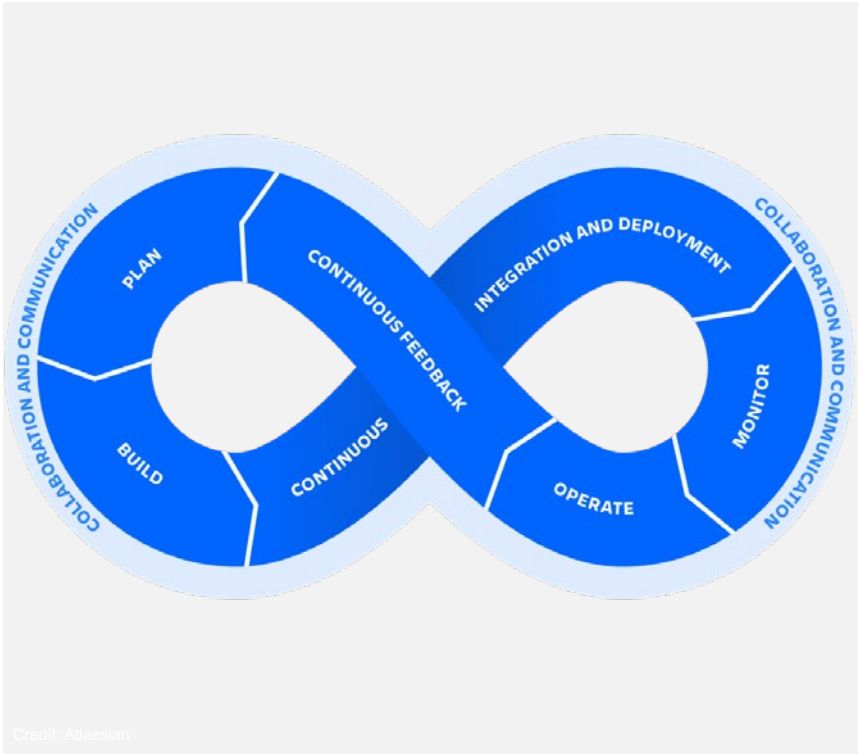


What is DevOps

And why does it matter

What is DevOps?

- Dev + Ops = Integration between devs and IT team
- Not only a technical matter, but a cultural shift (less siloes)
- Empower dev(s) to deploy to users



Why DevOps?

- Once set up it just works, and you can bypass the IT team
  - Automated process every time it’s triggered
  - Deploy to the right people
- Deploy fixes and new features more frequently in less time
  - Better, more advanced tools
  - More transparency with end user (credibility)
- Feel more confident <> increase reliability
  - Automated testing
  - Still needs human check, but reduces time

Speed

Improved collaboration

Rapid deployment

Quality and reliability

Security

DevOps process

Code

The developers work on smaller features (or bug fixes) on their branches.

Commit

While working on the codebase, the developers will frequently commit their code to a version control host.

When ready they will merge the branch.

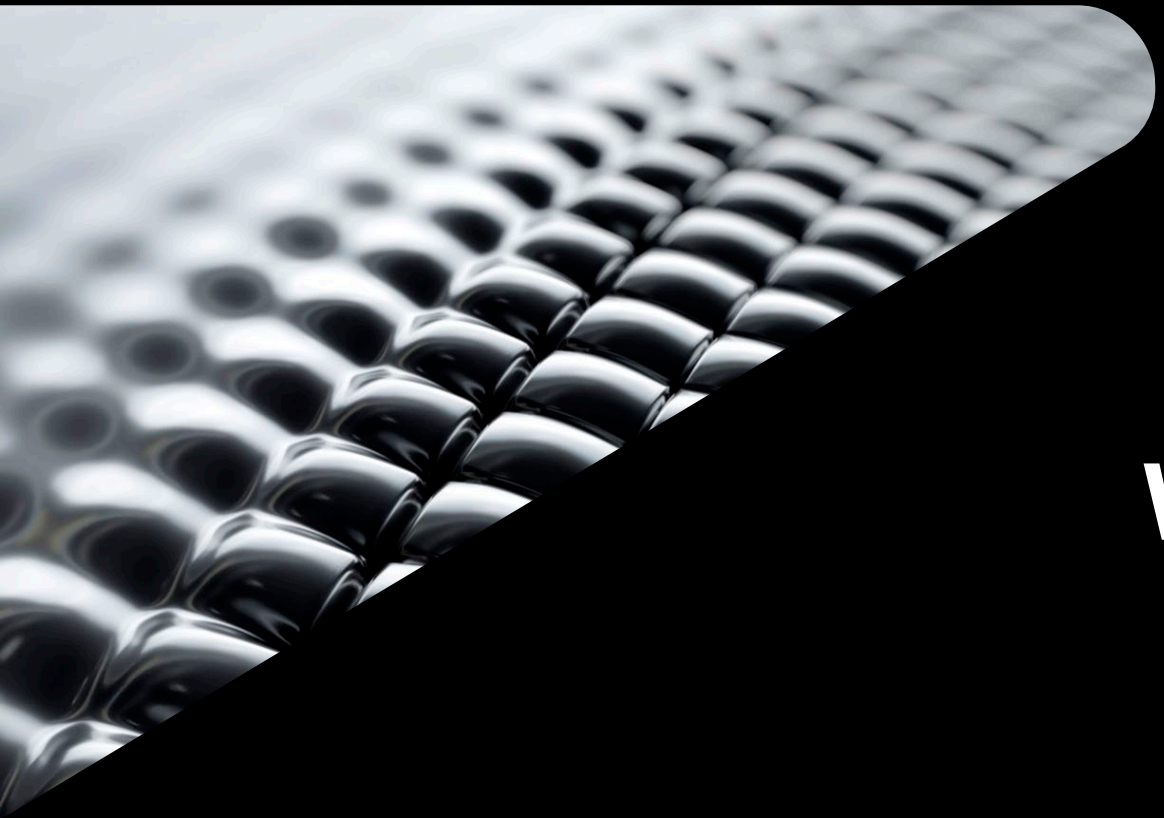
CI/CD

Commits or merges happening to a specific branch will trigger the pipeline to start.

This will start a process that will build the solution, including all the external and internal packages, it will version the .dll library, sign it and test it.

Deploy

If the previous steps went fine, the pipeline will deploy the necessary files to the right groups.



What is CI/CD

Continuous Integration / Continuous Delivery

The technical subset of DevOps

- Continuous Integration
  - Devs merge smaller code changes to central repo more often
  - Automated builds, tests and other actions are triggered
  - Find and address bugs quicker
  - Improve quality and reliability
  - Reduce release time and increase feedback
- Continuous Delivery
  - Deploy changes to test environment / group
  - Requires human decision to deploy to production
- Continuous Deployment
  - Completely automated deployment – doesn’t require human interaction
  - Not covered here

CI/CD Pipeline

Setup the pipeline

Install NuGets

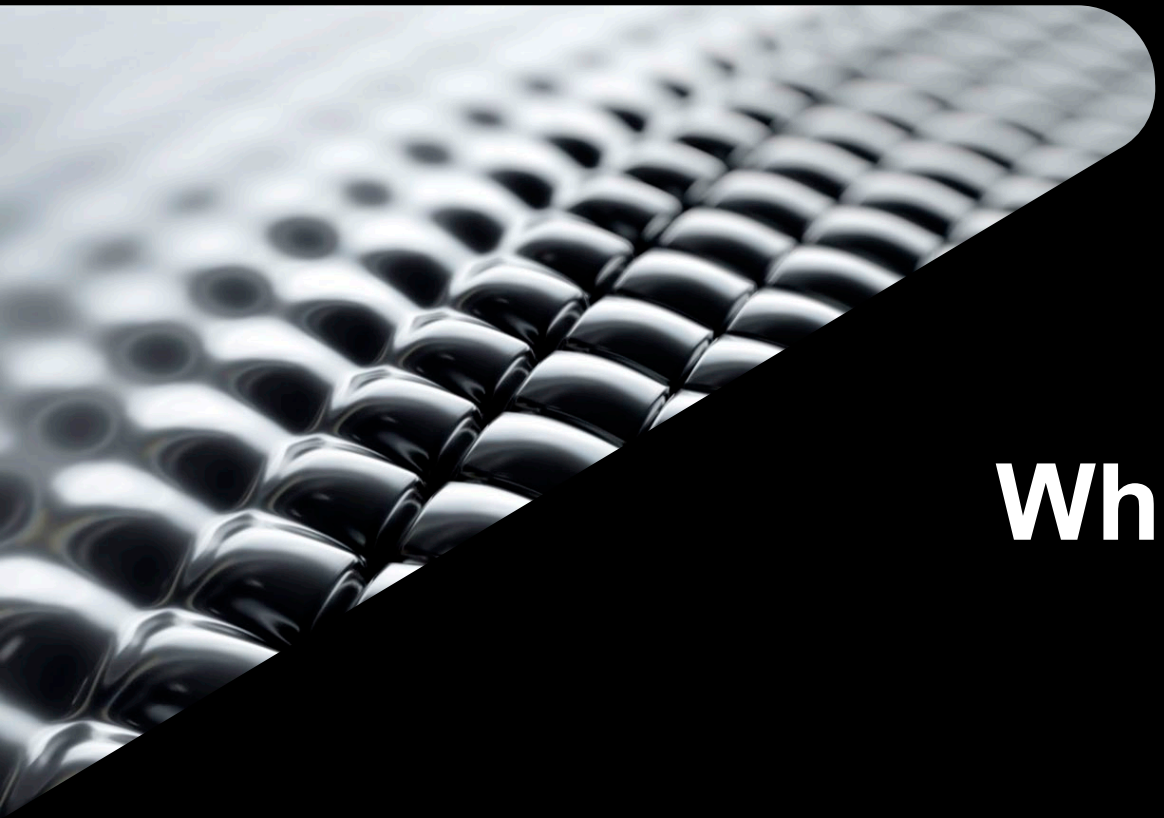
Build the solution

Version the files

Sign the files

Test the code

Deployment

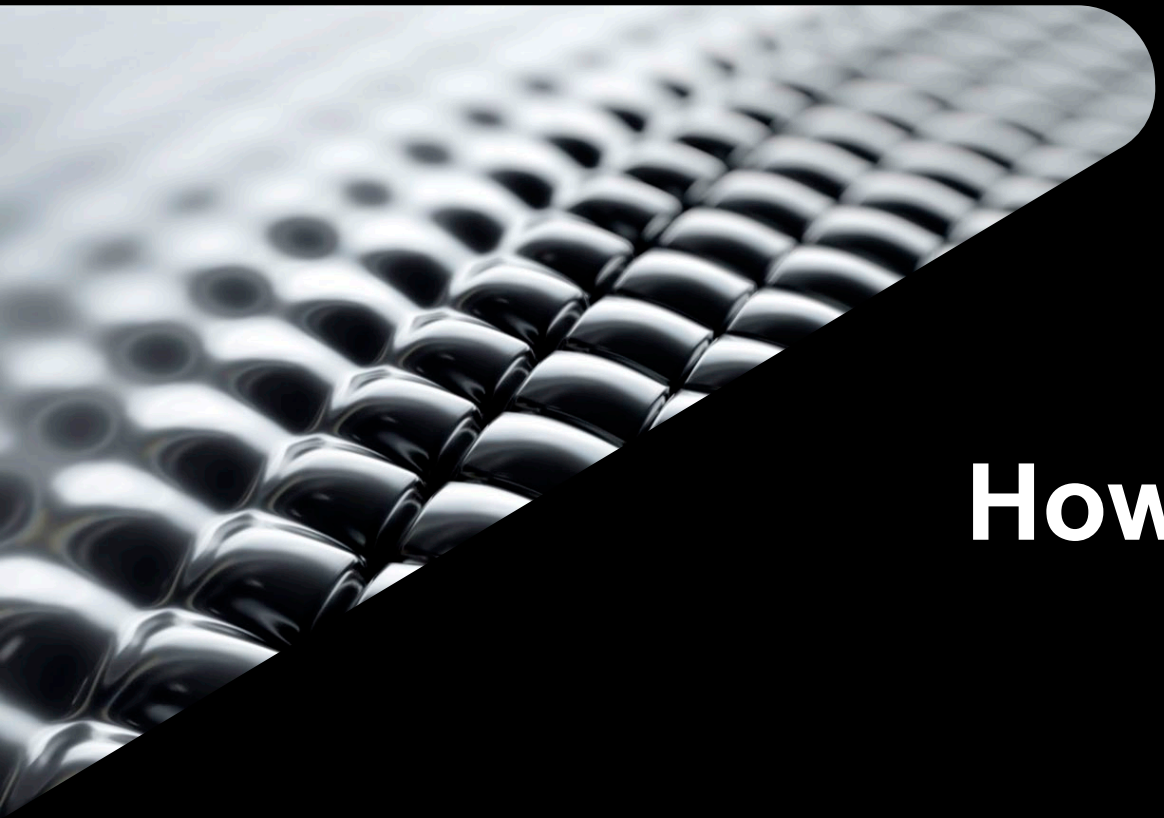


What you’ll need

Batteries not included

Tools and Accounts

1. Any remote git account (GitHub, BitBucket, etc.)
2. An Azure DevOps account (or AWS, etc.)
3. A self-hosted Windows virtual machine (VM) with Revit installed
4. A code certificate
5. pyRevit

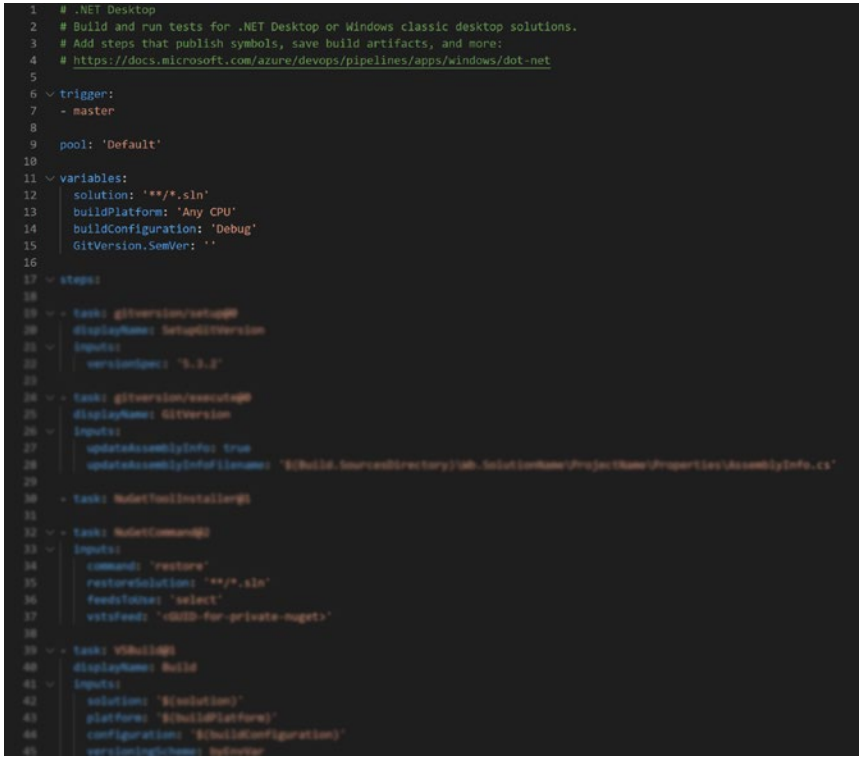


How to: overview

Practical bits

Pipeline Setup

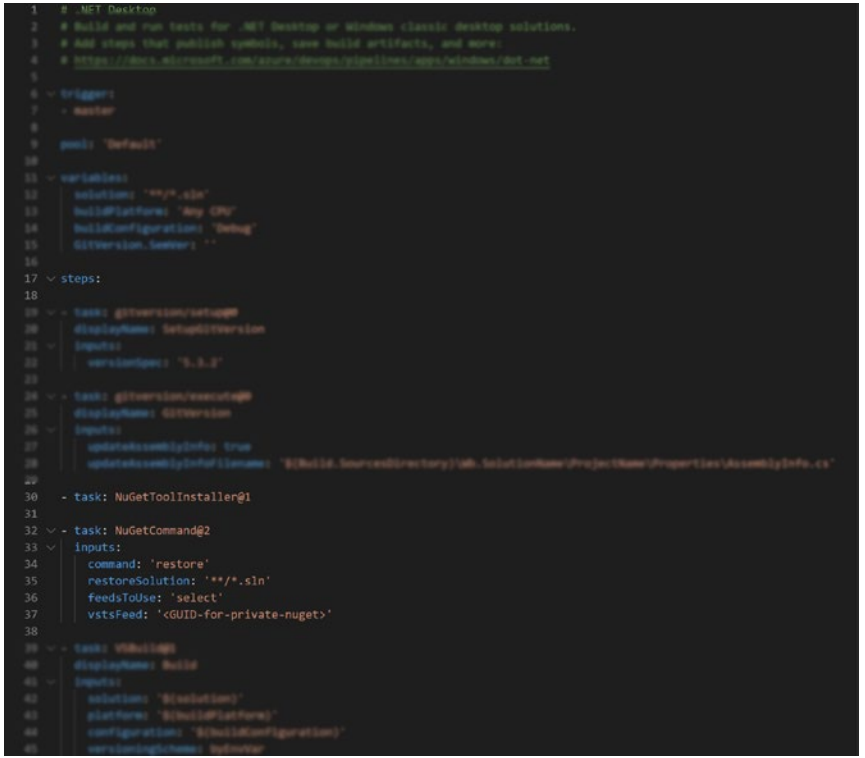
1. Create a new project in DevOps
2. In the new project, create a new pipeline
3. Connect the pipeline to your remote repo
4. Access your YAML file, automatically created by DevOps
5. Set the trigger (usually a commit / merge to a specific branch)
6. Define the machine (virtual or physical) that will run the pipeline. In our case you'll need a **self-hosted Windows VM** where you pre-install Revit!
7. Define the variables



NuGet Tasks

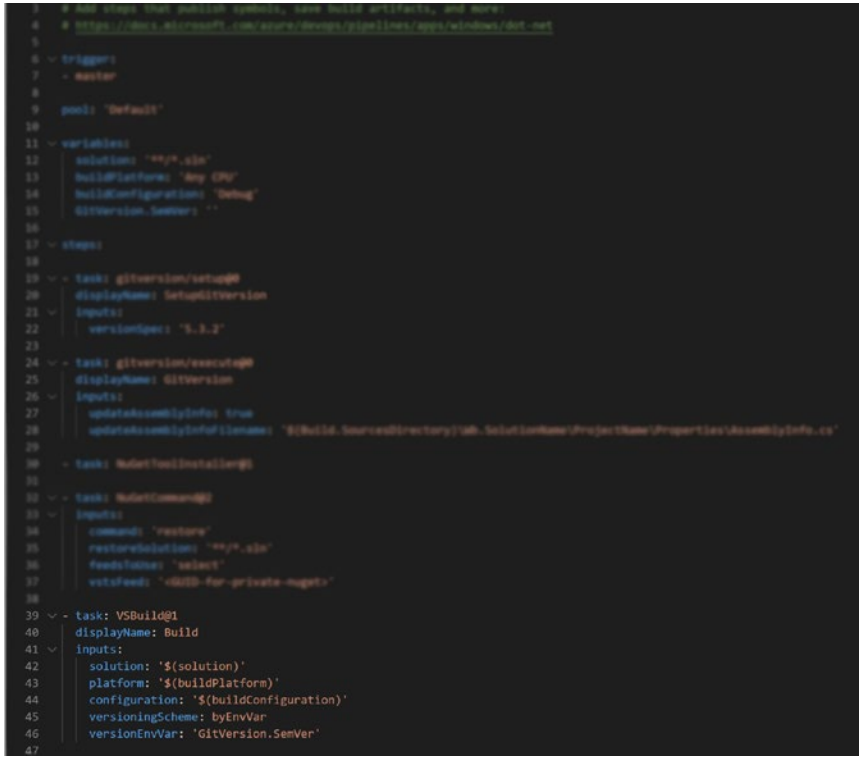
Get all used NuGet packages to successfully build solution

1. Install the NuGet tool in order to install all packages
2. Use the *restore* command to install external packages
3. Specify *vstsFeed* to get internal packages source



Build Solution

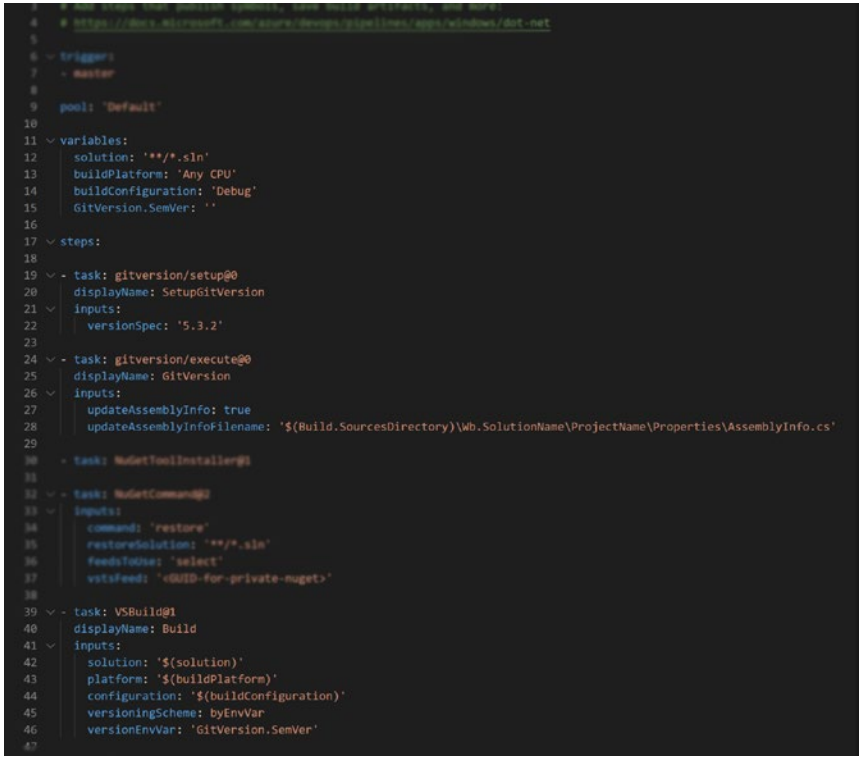
- Use MSBuild or VSBuild to build solution
- Most of the setup is done in the *csproj* and it's like building locally
- The Revit APIs are normally referenced from local files > the pipeline will likely fail. A solution is to use a NuGet package (we use ModPlus, but any will do)



```
# Add steps that publish symbols, save build artifacts, and more:
# https://docs.microsoft.com/azure/devops/pipelines/apps/windows/dot-net

trigger:
- master

pool: 'Default'

variables:
  solution: '**/*.sln'
  buildPlatform: 'Any CPU'
  buildConfiguration: 'Debug'
  GitVersion.SemVer: ''

steps:

- task: gitversion/setup@0
  displayName: SetupGitVersion
  inputs:
    versionSpec: '5.3.2'

- task: gitversion/execute@0
  displayName: GitVersion
  inputs:
    updateAssemblyInfo: true
    updateAssemblyInfoFilename: '$(Build.SourcesDirectory)\ab.SolutionName\ProjectName\Properties\AssemblyInfo.cs'

- task: NuGetToolInstaller@1

- task: NuGetCommand@2
  inputs:
    command: 'restore'
    restoreSolution: '**/*.sln'
    feedsToUse: 'select'
    vstsFeed: '<GUID-for-private-nuget>'

- task: VSBuild@1
  displayName: Build
  inputs:
    solution: '$(solution)'
    platform: '$(buildPlatform)'
    configuration: '$(buildConfiguration)'
    versioningScheme: byEnvVar
    versionEnvVar: 'GitVersion.SemVer'
```

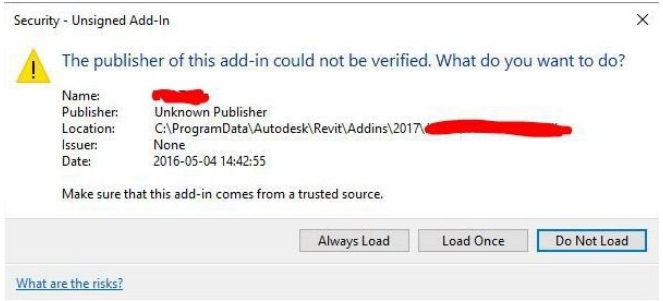
Semantic Versioning

1. This task will automatically assign a version to the resulting files based on the git history of the project
2. The repo needs at least a *main/master* and a *develop* branch not to fail
3. Add a *GitVersion.SemVer* variable to store the version info
4. Setup the *GitVersion* tool
5. Run *GitVersion* to calculate version
6. In the build task, apply the version to the built files



Code Signing

- To avoid unverified popup in Revit

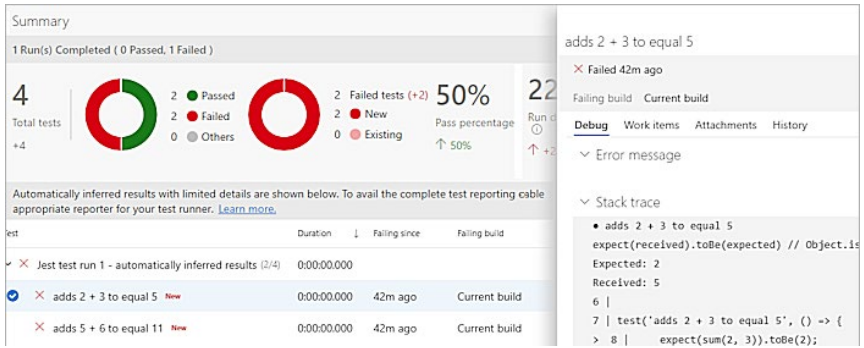


- Store your certificate either in the pipeline
- Or on Azure KeyVault
- The pipeline looks like the one on the right if you go the local route

```
    inputs:
      solution: '$(solution)'
      platform: '$(buildPlatform)'
      configuration: '$(buildConfiguration)'
      versioningScheme: byEnvVar
      versionEnvVar: 'GitVersion.SemVer'

  # Sign Revit dll
  - task: codesigning@2
    displayName: CodeSigning
    inputs:
      secureFileId: 'NameOfCertificate'
      signCertPassword: 'CertificatePassword'
      files: '$(Build.SourcesDirectory)\Path\To\Dlls.dll'
      timeServer: 'http://timestamp.digicert.com'
      hashingAlgorithm: 'SHA256'

  - task: PowerShell@2
    displayName: PowershellRunRevitTests
    inputs:
      targetType: 'inline'
      script: |
        cd ..\..\..\..\
        cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug\
        .\RevitTestFrameworkConsole.exe --dir Path\To\Framework -a Path\To\Assembly\with\tests.dll -r Path\

  - task: PublishTestResults@2
    inputs:
      testResultsFormat: 'NUnit'
      testResultsFiles: '**/results.xml'
      failTaskOnFailedTests: true

  - task: PowerShell@2
    displayName: PowershellPushToWombat
    inputs:
      targetType: 'inline'
      script: |
        [string]$RepoDirectory = 'Path\To\Repo\Bin\Folder\On\Current\Machine'
        [string]$AzureDirectory  = 'Path\To\Built\Dlls'
        cd ..\..\..\..\
```

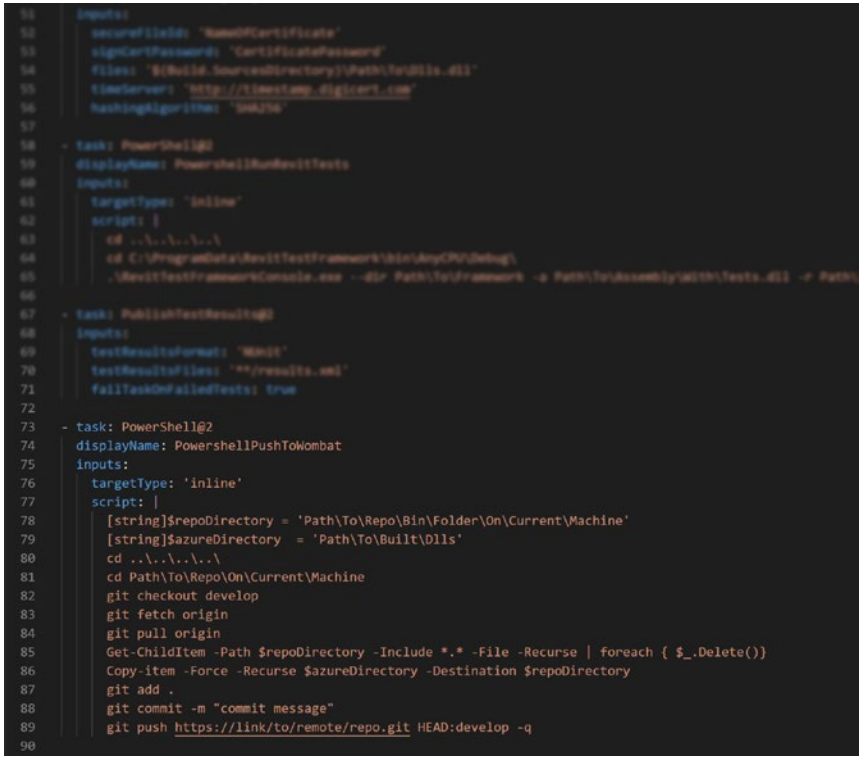
Automated (Unit / Integration) Testing

- The tests live under a separate project in the same solution
- We use Dynamo's RevitTestFramework to run the tests, but you have options
- After building, the Windows VM with Revit installed will open Revit and run the tests
- The results of the tests will be published in the pipeline and stop the execution in case one or more tests failed



Deployment

- Our Wombat toolbar for Revit is built upon **pyRevit**, which makes deployment convenient, being based on git
- The task is running a few git commands in order to reach all our test groups and general users
- Process
  1. *Checkout* the correct branch to deploy to
  2. *Pull* from origin to ensure branch is up-to-date
  3. Get all files to deploy from build
  4. *Add* files to commit
  5. *Commit* with meaningful repeated message
  6. *Push* to repository
- pyRevit will do the rest for you!



CI/CD Pipeline

Setup the pipeline

Install NuGets

Build the solution

Version the files

Sign the files

Test the code

Deployment

Conclusion

If you fell asleep halfway and only bring this home

Key Takeaways

If you forget everything else but this

Better process: DevOps practices, through the help of CI/CD **standardization** will save time and reduce bugs, hence saving money

Better collaboration: working together gets smoother and less stressful, by establishing a **controlled, reliable process** (less bugs deployed)

Better results: not only by deploying tools that are more reliable, but also by releasing more often and **increasing feedback**


THANK YOU
FOR LISTENING
imgflip.com