



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

SD501013

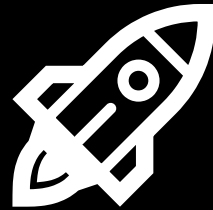
Andrea Tassera

Global Specialist Design Technology - Woods Bagot

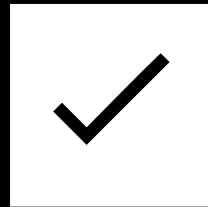
@dre_tas



**How long does it
take you to deploy
tools?**



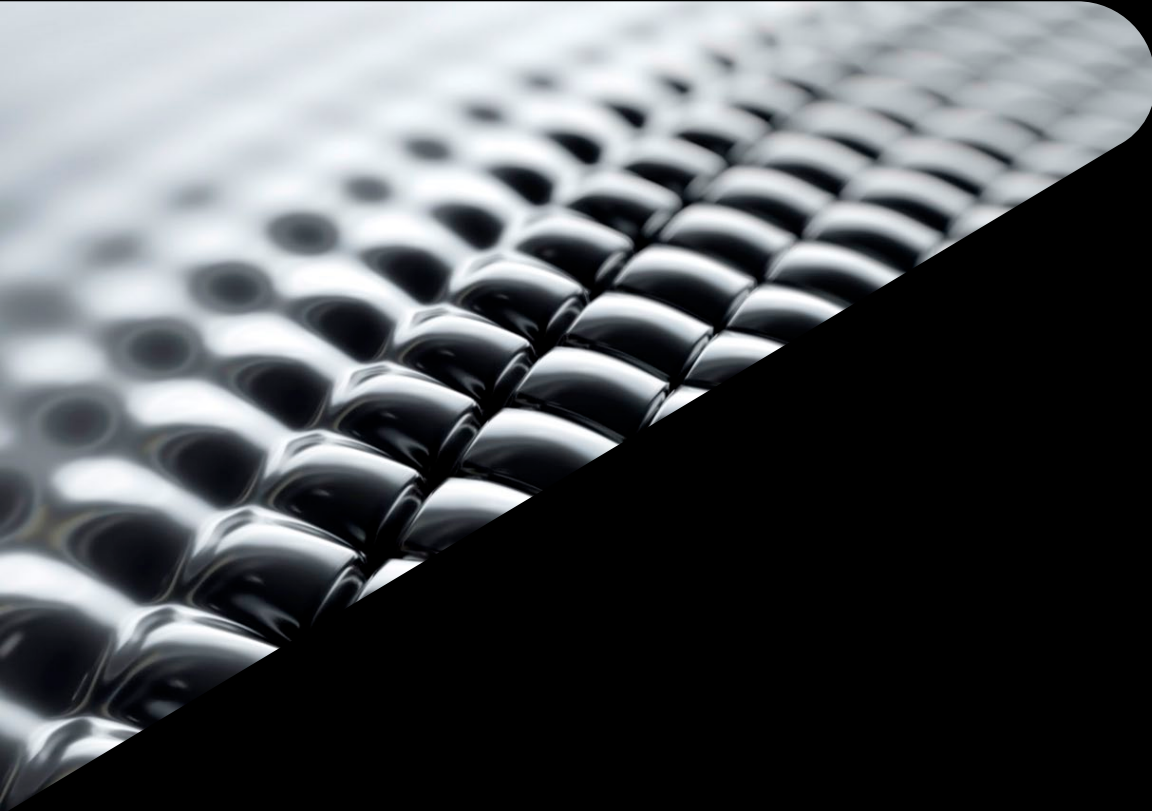
**How do you ensure
quality?**



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.

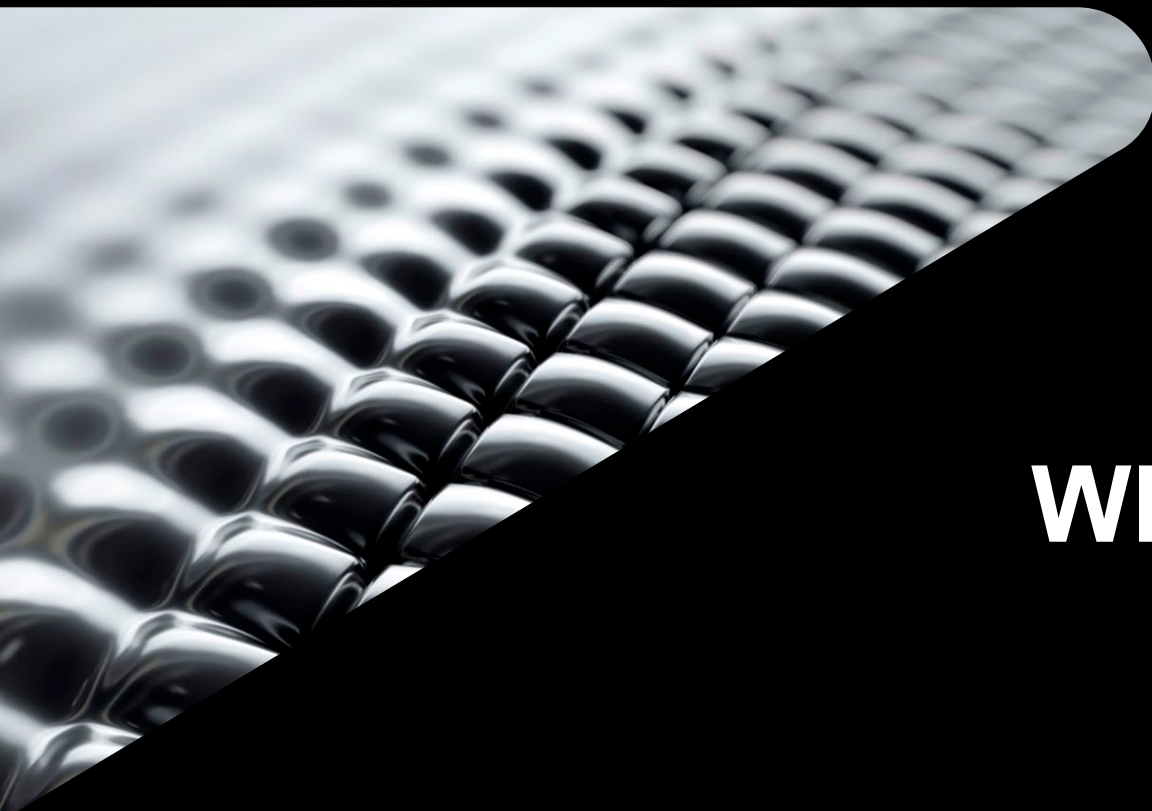




Assumptions

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)

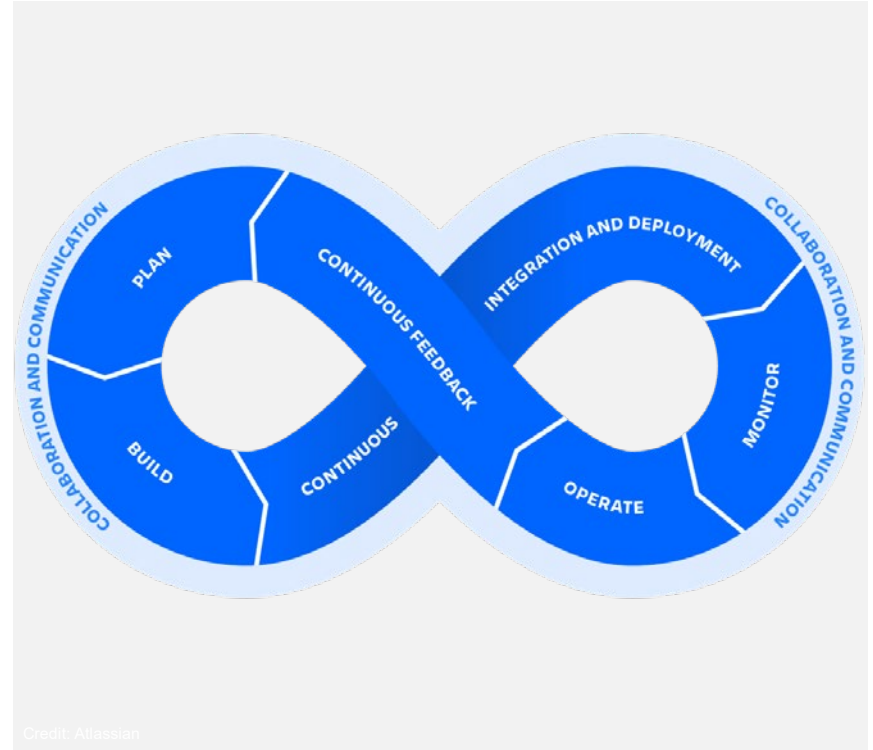


What is DevOps

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.

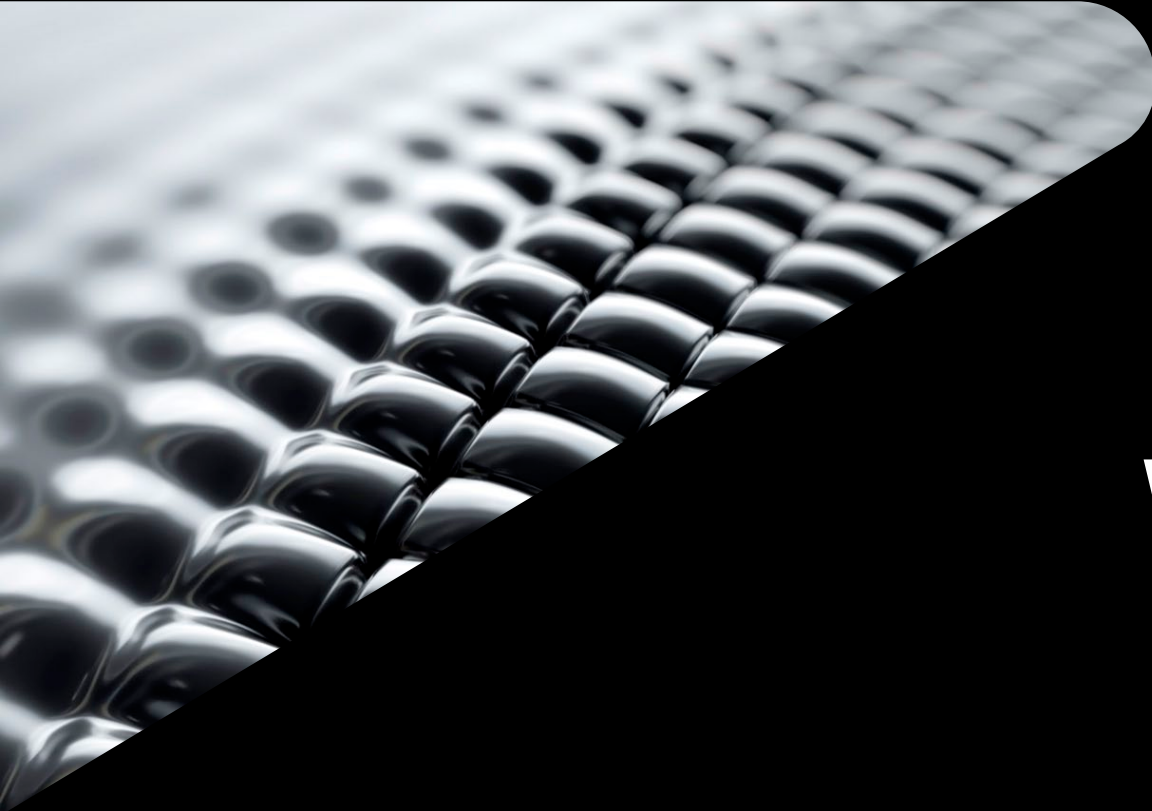
CI/CD

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



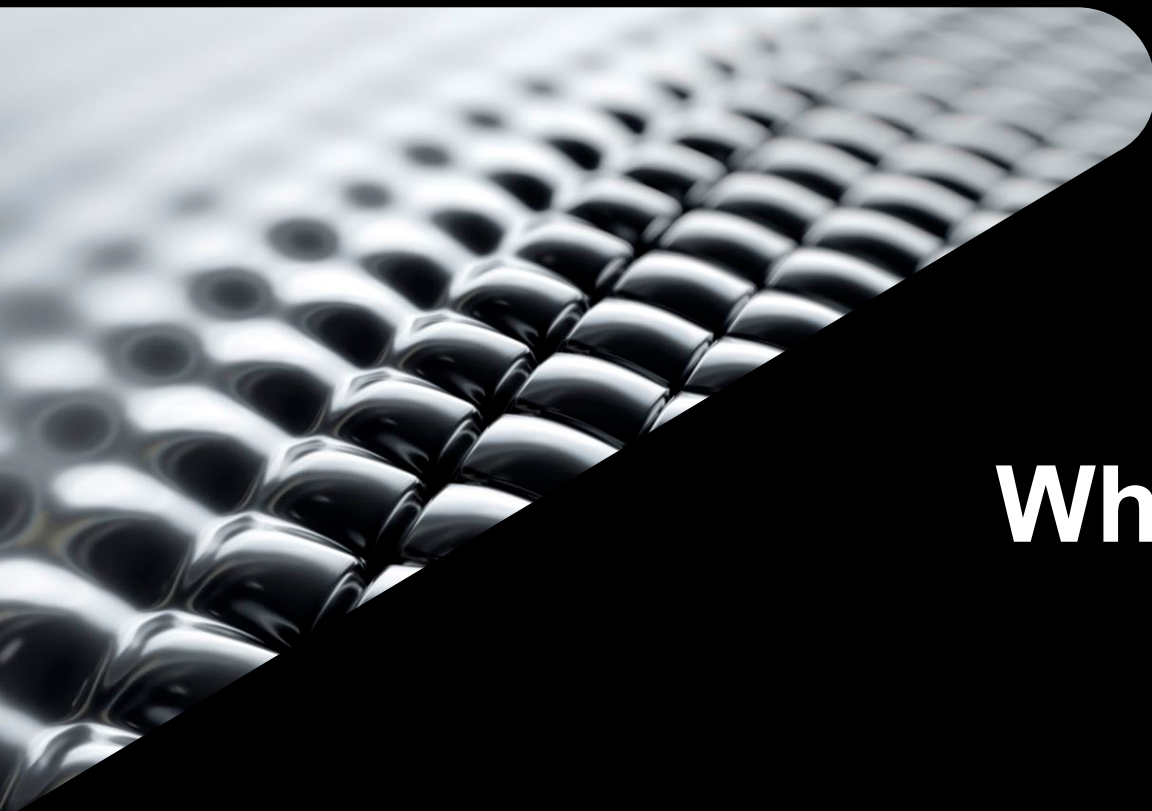
Sign the files



Test the code



Deployment

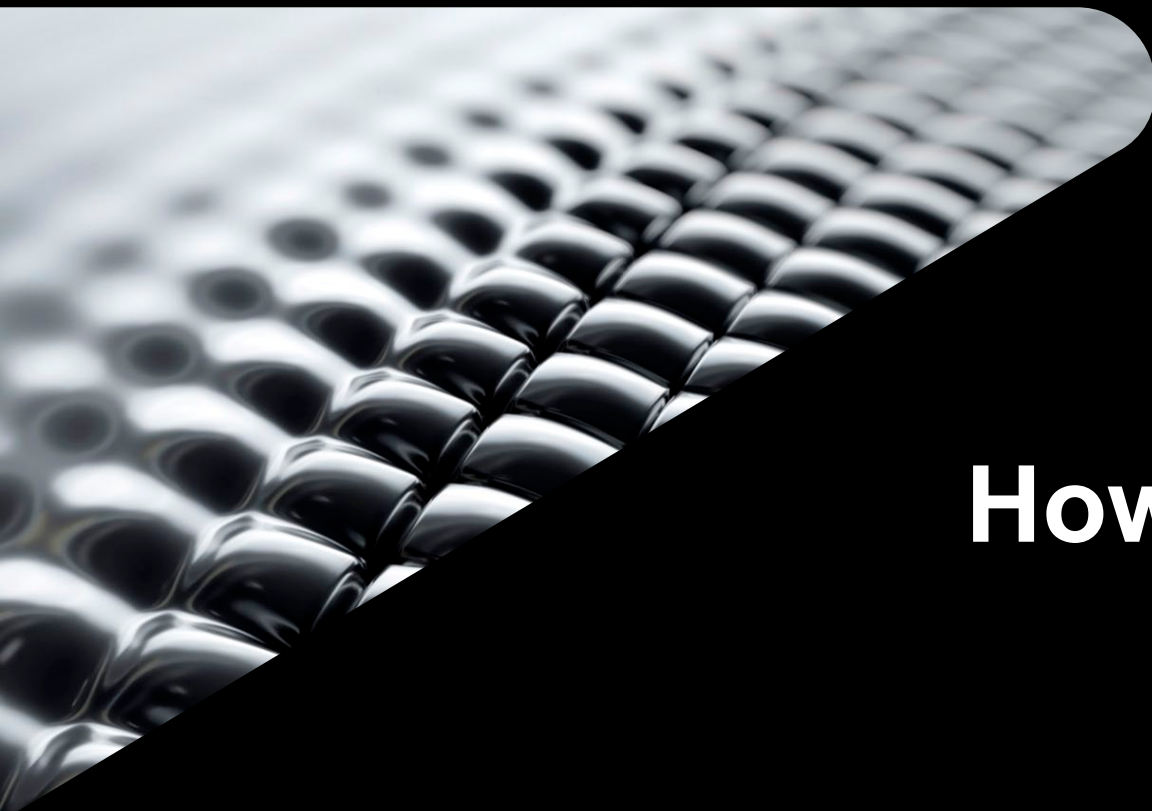


What you'll need

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

```
1 # .NET Desktop
2 # Build and run tests for .NET Desktop or Windows classic desktop solutions.
3 # Add steps that publish symbols, save build artifacts, and more:
4 # https://docs.microsoft.com/azure/devops/pipelines/apps/windows/dot-net
5
6 trigger:
7   - master
8
9 pool: 'Default'
10
11 variables:
12   solution: '**/*.sln'
13   buildPlatform: 'Any CPU'
14   buildConfiguration: 'Debug'
15   GitVersion.SemVer: ''
16
17 steps:
18   - task: gitversion/setup@0
19     displayName: SetupGitVersion
20     inputs:
21       versionSpec: '5.3.2'
22
23   - task: gitversion/execute@0
24     displayName: GitVersion
25     inputs:
26       updateAssemblyInfo: true
27       updateAssemblyInfoFilename: '$(Build.SourcesDirectory)/obj/$(SolutionName)/$(ProjectName)/Properties/AssemblyInfo.cs'
28
29   - task: NuGetToolInstaller@0
30
31   - task: NuGetCommand@0
32     inputs:
33       command: 'restore'
34       restoreSolution: '**/*.sln'
35       feedSources: 'select'
36       vstsFeed: 'vsssf-private-nuget'
37
38   - task: VSBuild@0
39     displayName: Build
40     inputs:
41       solution: '$(solution)'
42       platform: '$(buildPlatform)'
43       configuration: '$(buildConfiguration)'
44       versionFolderPath: subfolder
```

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

```
1 # .NET Desktop
2 # Build and run tests for .NET Desktop or Windows classic desktop solutions.
3 # Add steps that publish symbols, save build artifacts, and more:
4 # https://docs.microsoft.com/azure/devops/pipelines/apps/windows/dot-net
5
6 triggers:
7 - master
8
9 pool: 'Default'
10
11 variables:
12   solution: '**/*.sln'
13   buildPlatform: 'Any CPU'
14   buildConfiguration: 'Debug'
15   dotNetVersion: ''
16
17 steps:
18
19 - task: GetVersion/setup@0
20   displayName: SetupGetVersion
21   inputs:
22     versionSpec: '5.3.0'
23
24 - task: GetVersion/execute@0
25   displayName: GetVersion
26   inputs:
27     updateAssemblyInfo: true
28     updateAssemblyInfoFileName: '$(Build.SourcesDirectory)/obj/$(SolutionName)/$(ProjectName)/Properties/AssemblyInfo.cs'
29
30 - task: NuGetToolInstaller@1
31
32 - task: NuGetCommand@2
33   inputs:
34     command: 'restore'
35     restoreSolution: '**/*.sln'
36     feedsToUse: 'select'
37     vstsFeed: 'cGUID-for-private-nuget'
38
39 - task: VSTest@0
40   displayName: Build
41   inputs:
42     solution: '$(solution)'
43     platform: '$(buildPlatform)'
44     configuration: '$(buildConfiguration)'
45     versionSpec: ''
```

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)

```
3 # Add steps that publish symbols, save build artifacts, and more:
4 # https://docs.microsoft.com/en-us/azure/devops/pipelines/apps/windows/dot-net
5
6 - triggers:
7   - master
8
9 pool: 'Default'
10
11 variables:
12   solution: '**/*.sln'
13   buildPlatform: 'Any CPU'
14   buildConfiguration: 'Debug'
15   GitVersion.SemVer: ''
16
17 stages:
18
19   - task: gitversion/setup@0
20     displayName: SetupGitVersion
21     inputs:
22       versionSpec: '5.3.2'
23
24   - task: gitversion/execute@0
25     displayName: GitVersion
26     inputs:
27       updateAssemblyInfo: true
28       updateAssemblyInfoFilename: '$(Build.SourcesDirectory)/$(Build.SolutionName)/$(ProjectName)/Properties/AssemblyInfo.cs'
29
30   - task: NuGetToolInstaller@0
31
32   - task: NuGetCommand@0
33     inputs:
34       commands: 'restore'
35       restoreSolution: '**/*.sln'
36       feedsFolder: 'select'
37       vstsFeed: 'vssd-for-private-nugets'
38
39   - task: VSBuild@1
40     displayName: Build
41     inputs:
42       solution: '$(solution)'
43       platform: '$(buildPlatform)'
44       configuration: '$(buildConfiguration)'
45       versioningScheme: byEnvVar
46       versionEnvVar: 'GitVersion.SemVer'
47
```

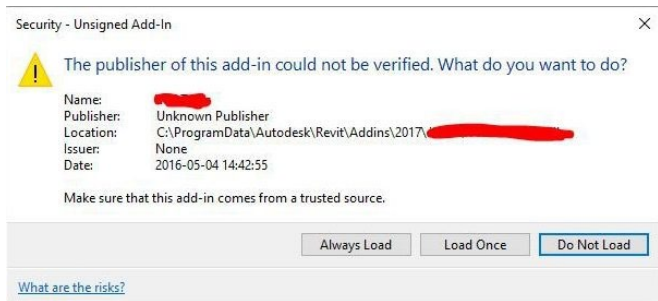
Semantic Versioning

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

```
1 # Add triggers from GitHub (optional), Azure DevOps, and more
2 # https://docs.microsoft.com/en-us/azure/devops/pipelines/apps/windows/dot-net
3
4 - trigger:
5   - master
6
7 pool: 'Default'
8
9 variables:
10   solution: '**/*.sln'
11   buildPlatform: 'Any CPU'
12   buildConfiguration: 'Debug'
13   GitVersion.SemVer: ''
14
15 steps:
16
17   - task: gitversion/setup@0
18     displayName: SetupGitVersion
19     inputs:
20       versionSpec: '5.3.2'
21
22   - task: gitversion/execute@0
23     displayName: GitVersion
24     inputs:
25       updateAssemblyInfo: true
26       updateAssemblyInfoFilename: '$(Build.SourcesDirectory)\Web.SolutionName\ProjectName\Properties\AssemblyInfo.cs'
27
28   - task: NuGetToolInstaller@1
29
30   - task: NuGetCommand@2
31     inputs:
32       command: 'restore'
33       restoreSolution: '**/*.sln'
34       feedsToUse: 'select'
35       vstsFeed: '<GUD-for-private-nuget>'
36
37   - task: VSBUILD@1
38     displayName: Build
39     inputs:
40       solution: '$(solution)'
41       platform: '$(buildPlatform)'
42       configuration: '$(buildConfiguration)'
43       versioningScheme: 'byEnvVar'
44       versionEnvVar: 'GitVersion.SemVer'
45
46
```

Code Signing

- To avoid unverified popup in Revit

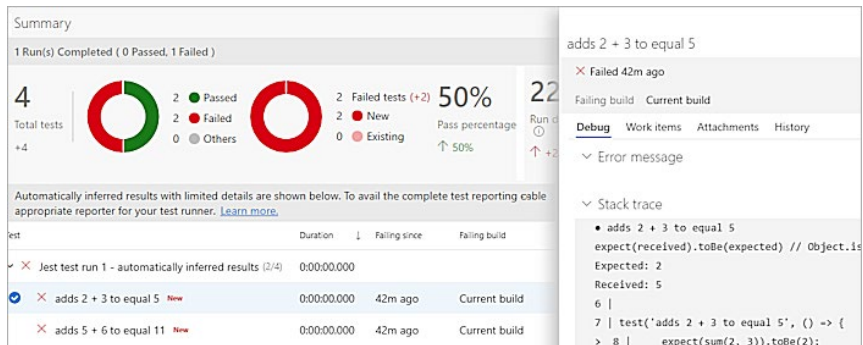


- Store your certificate either in the pipeline
- Or on Azure KeyVault
- The pipeline looks like the one on the right if you go the local route

```
41 inputs:
42   solution: '$(solution)'
43   platform: '$(buildPlatform)'
44   configuration: '$(buildConfiguration)'
45   versioningScheme: 'byFolder'
46   versionSuffix: '$(Version.Suffix)'
47
48 # Sign Revit dll
49 - task: codesigning@2
50   displayName: CodeSigning
51   inputs:
52     secureFileId: 'NameOfCertificate'
53     signCertPassword: 'CertificatePassword'
54     files: '$(Build.SourcesDirectory)\Path\To\Dlls.dll'
55     timeServer: 'http://timestamp.digicert.com'
56     hashingAlgorithm: 'SHA256'
57
58 - task: PowerShell@2
59   displayName: PowerShellRunRevitTests
60   inputs:
61     targetType: 'inline'
62     scripts: |
63       cd ..\..\..\
64       cd C:\ProgramData\Autodesk\Revit\Addins\2017\AnyOf\Debug\
65       .\RevitTestFrameworkConsole.exe --dir Path\to\framework --path Path\to\assembly\with\tests.dll --path
66
67 - task: PublishTestResults@2
68   inputs:
69     testResultFormat: 'JUnit'
70     testResultFiles: '**\results.xml'
71     failTaskOnFailedTests: true
72
73 - task: PowerShell@2
74   displayName: PowerShellPushToAzure
75   inputs:
76     targetType: 'inline'
77     scripts: |
78       [string]$srcDirectory = "Path\to\Repo\bin\folder\bin\Current\Machine"
79       [string]$destDirectory = "Path\to\bin\Dlls"
80       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



```

12  outputs:
13    - outputPath: 'Results.xml'
14    - displayName: 'BuildTestResults'
15    configurations: '$(BuildConfiguration)'
16    versioningScheme: 'None'
17    versionNumber: '$(BuildNumber)'
18
19 # Page Result dll
20 tasks:
21   - task: Publishing@1
22     displayName: 'Publishing'
23     inputs:
24       sourcePath: '$(BuildSourceFolder)'
25       targetPath: '$(BuildTargetFolder)'
26       files: '$(BuildSourceFolder)\*.xml'
27       fileName: '$(BuildSourceFolder)\*.xml'
28       fileName: '$(BuildSourceFolder)\*.xml'
29       fileName: '$(BuildSourceFolder)\*.xml'
30       fileName: '$(BuildSourceFolder)\*.xml'
31
32 - task: PowerShell@2
33   displayName: 'PowerShellRunRevitTests'
34   inputs:
35     targetType: 'inline'
36     script: |
37       cd $(BuildSourceFolder)
38       cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug
39       .\RevitTestFrameworkConsole.exe -dir PathToFramework -a PathToAssemblyWithTests.dll -r PathToResults.xml -revit "C:\Program Files\Autodesk\Revit 2020\Revit.exe" --continue
40
41 - task: PublishTestResults@2
42   inputs:
43     testResultsFormat: 'JUnit'
44     testResultsFiles: '**\results.xml'
45     failTaskOnFailedTests: true
46
47 tasks:
48   - task: PowerShell@2
49     displayName: 'PowerShellRunRevitTests'
50     inputs:
51       targetType: 'inline'
52       script: |
53         cd $(BuildSourceFolder)
54         cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug
55         .\RevitTestFrameworkConsole.exe -dir PathToFramework -a PathToAssemblyWithTests.dll -r PathToResults.xml -revit "C:\Program Files\Autodesk\Revit 2020\Revit.exe" --continue
56
57   - task: PowerShell@2
58     displayName: 'PowerShellRunRevitTests'
59     inputs:
60       targetType: 'inline'
61       script: |
62         cd $(BuildSourceFolder)
63         cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug
64         .\RevitTestFrameworkConsole.exe -dir PathToFramework -a PathToAssemblyWithTests.dll -r PathToResults.xml -revit "C:\Program Files\Autodesk\Revit 2020\Revit.exe" --continue
65
66   - task: PowerShell@2
67     displayName: 'PowerShellRunRevitTests'
68     inputs:
69       targetType: 'inline'
70       script: |
71         cd $(BuildSourceFolder)
72         cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug
73         .\RevitTestFrameworkConsole.exe -dir PathToFramework -a PathToAssemblyWithTests.dll -r PathToResults.xml -revit "C:\Program Files\Autodesk\Revit 2020\Revit.exe" --continue
74
75   - task: PowerShell@2
76     displayName: 'PowerShellRunRevitTests'
77     inputs:
78       targetType: 'inline'
79       script: |
80         cd $(BuildSourceFolder)
81         cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug
82         .\RevitTestFrameworkConsole.exe -dir PathToFramework -a PathToAssemblyWithTests.dll -r PathToResults.xml -revit "C:\Program Files\Autodesk\Revit 2020\Revit.exe" --continue
83
84   - task: PowerShell@2
85     displayName: 'PowerShellRunRevitTests'
86     inputs:
87       targetType: 'inline'
88       script: |
89         cd $(BuildSourceFolder)
90         cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug
91         .\RevitTestFrameworkConsole.exe -dir PathToFramework -a PathToAssemblyWithTests.dll -r PathToResults.xml -revit "C:\Program Files\Autodesk\Revit 2020\Revit.exe" --continue
92
93   - task: PowerShell@2
94     displayName: 'PowerShellRunRevitTests'
95     inputs:
96       targetType: 'inline'
97       script: |
98         cd $(BuildSourceFolder)
99         cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug
100        .\RevitTestFrameworkConsole.exe -dir PathToFramework -a PathToAssemblyWithTests.dll -r PathToResults.xml -revit "C:\Program Files\Autodesk\Revit 2020\Revit.exe" --continue

```

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!

```
51 inputs:
52   secureFileId: "NameOfCertificate"
53   signCertPassword: "CertificatePassword"
54   files: "$(Build.SourcesDirectory)\PathToDlls.dll"
55   fileServer: "http://filestamp.digicert.com"
56   hashingAlgorithm: "sha256"
57
58 - task: PowerShell@2
59   displayName: PowershellRunRevitTests
60   inputs:
61     targetType: 'inline'
62     script: |
63       cd ..\..\..\
64       cd C:\ProgramData\RevitTestFramework\bin\AnyCPU\Debug\
65       .\RevitTestFrameworkConsole.exe --dir PathToFramework -a PathToAssembly\PathToTests.dll -p Path
66
67 - task: PublishTestResults@2
68   inputs:
69     testResultsFormat: 'JUnit'
70     testResultsFiles: "**/results.xml"
71     failTaskOnFailedTests: true
72
73 - task: PowerShell@2
74   displayName: PowershellPushToWombat
75   inputs:
76     targetType: 'inline'
77     script: |
78       [string]$repoDirectory = 'Path\To\Repo\Bin\Folder\On\Current\Machine'
79       [string]$azureDirectory = 'Path\To\Built\Dlls'
80       cd ..\..\..\
81       cd Path\To\Repo\On\Current\Machine
82       git checkout develop
83       git fetch origin
84       git pull origin
85       Get-ChildItem -Path $repoDirectory -Include *.* -File -Recurse | foreach { $_.Delete() }
86       Copy-Item -Force -Recurse $azureDirectory -Destination $repoDirectory
87       git add .
88       git commit -m "commit message"
89       git push https://link/to/remote/repo.git HEAD:develop -q
90
```

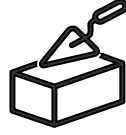
CI/CD Pipeline



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



Autodesk and the Autodesk logo are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product offerings, specifications and pricing at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document.

© 2022 Autodesk. All rights reserved.