

DevOps per le applicazioni desktop

Matteo Pagani

Windows App Consult Engineer @ Microsoft
matteo.pagani@microsoft.com | @qmatteoq

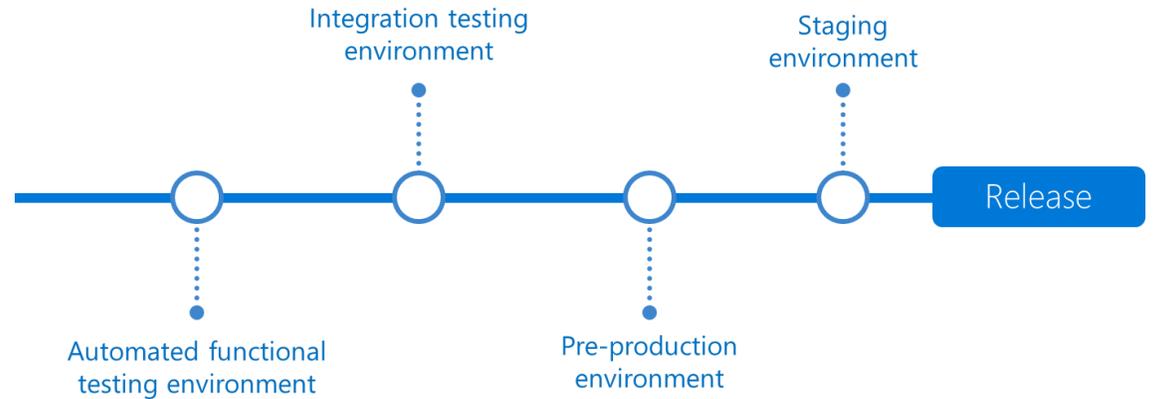
Container &
DevOps Day

DevOps for Windows desktop apps

101010
010101
101010

Continuous Deployment (CD)

- By combining continuous integration and infrastructure as code (IaC), you'll achieve identical deployments and the confidence to deploy to production at any time.
- With continuous deployment, you can automate the entire process from code commit to production if your CI/CD tests are successful.



Demo

The end-to-end workflow

Container &
DevOps Day

DevOps for Desktop Apps



Key Benefits of MSIX



Modern IT Virtuous Cycle

- Enable OS, Apps & IT to update independently

Never Regret Installing an app

- Predictable, safe, and reliable deployment
- Trusted, clean install & uninstall – no Win ROT

Simpler packaging and deployment

- Declarative install via Manifest file
- Identity, formal versioning and device targeting

Optimized

- Disk space
- Network optimization using differential updates

Packaging with MSIX

MSIX is the package format to deploy ANY Windows application

MFC, Win32, UWP, .NET Framework, .NET Core 3... or any other EXE

Publishing Anywhere

Managed: Store, Intune, SCCM

Unmanaged: Custom locations (Web, UNC)

with App Installer app and .appinstaller file

Support .NET Core / .NET 5 deployment modes

Self Contained, Framework Dependent

Create MSIX with Visual Studio

Windows Application Packaging Project (.wapproj)

Manifest Editor

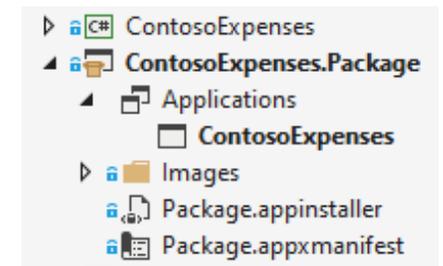
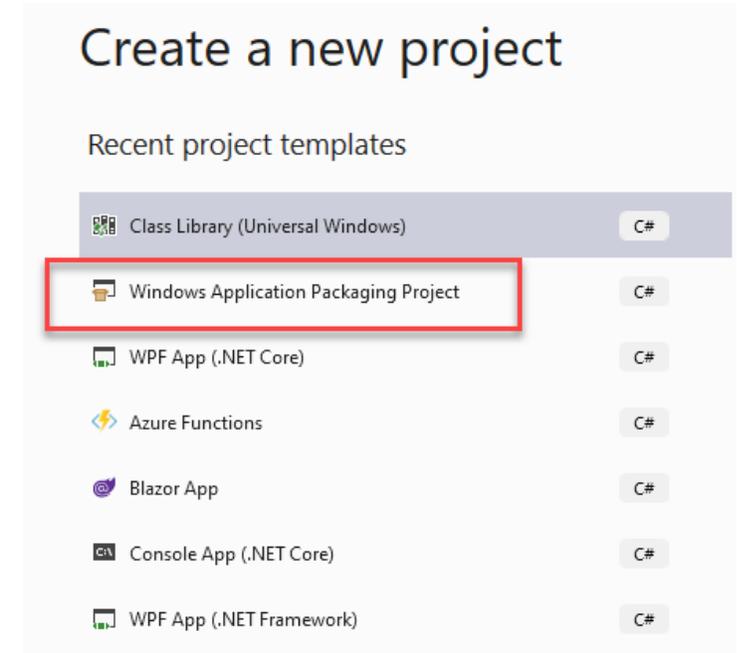
Debugging

Packages Wizard

MSBuild based

Customize build/package with MSBuild properties

Available in the UWP Workload



Demo

Add MSIX packaging to a .NET application

Container &
DevOps Day

MSIX Extensions for Azure DevOps

- It simplifies setting all the parameters you need to generate the MSIX package as part of the pipeline
- Package not just Visual Studio projects, but any binary
- Takes care of updating the version number in the manifest

MSIX build and package ⓘ [Link settings](#) [View YAML](#) [Remove](#)

Task version

Display name *

Output Path * ⓘ

Build Solution with MSBuild ⓘ

Project to Build * ⓘ ...

Clean Before Building ⓘ

Generate MSIX Bundle ⓘ

Configuration * ⓘ

Platform * ⓘ

Update App Version in Manifest ⓘ

Manifest File to Update Version * ⓘ ...

App Version * ⓘ

Application Package Distribution Mode * ⓘ

Advanced Options for MSBuild ▾

Control Options ▾

Output Variables ▾

Builds

Hosted Build Agent

Include all VS features

Can be extended with new SDKs

Run automated tests

Clean images for each build

Private Build Agent

Access local resources (UNC shares)

Speed up build times

Custom SDKs installed in advance

Security with public PRs

- GitHub and Azure DevOps provide best-in class support for building Windows apps
- Use **windows-latest** on Azure DevOps and GitHub

Demo

Configuring CI/CD

Container &
DevOps Day

Release Pipelines

Releases deploy artifacts to environments

Client apps should be signed at release time

Promotion based on approval workflow

The screenshot displays a release pipeline interface. On the left, a sidebar shows 'Releases' and 'Artifacts' sections. The main area shows a release triggered via push 5 minutes ago by user qmatteoq. The release is currently 'In progress'. Below this, a pipeline named 'azure-static-web-apps-white-wave-0acbb4a03.yml' is shown, triggered on push. The pipeline consists of three steps: 'build' (2m 44s, completed), 'Deploy to staging envir...' (1m 41s, completed), and 'Deploy to production envir...' (11s, in progress). The 'Deploy to production' step is currently showing 'Deploying to production'.

Triggered via push	Status	Total duration	Artifacts
5 minutes ago qmatteoq pushed <code>092ae7b</code> <code>main</code>	In progress	—	—

azure-static-web-apps-white-wave-0acbb4a03.yml
on: push

- build 2m 44s [Completed]
- Deploy to staging envir... 1m 41s [Completed]
- Deploy to production envir... 11s [In Progress]
Deploying to production

Signing

CA issued certificates

Public CA's enable wide distribution

Private CA's enable enterprise distribution

Self Signed certificates

Development/testing purposes only

Not for wide distribution

MSIX extensions

- Task for Azure DevOps to sign any kind of file, including MSIX packages
- Based on the signtool utility included in the Windows 10 SDK
- Supports Secure Files, to help you keeping the private certificate safe



[Link](#)

MSIX package signing ⓘ

Task version ▾

Display name *

Package to sign * ⓘ

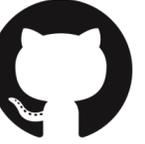
Certificate File * ⓘ

Password Variable * ⓘ

Time Stamp Server ⓘ

Control Options ▾

Output Variables ▾



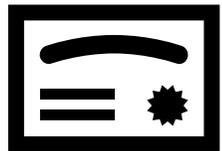
Code signing on GitHub

- Use a PowerShell script to encode the certificate in base64
- Store it as a secret, together with the password
- Download the certificate as part of the build and use it to sign the package
- Delete the certificate at the end

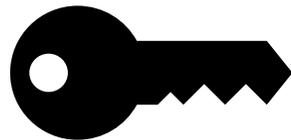
The starter workflow on GitHub for .NET Core desktop apps uses this approach: <https://devblogs.microsoft.com/dotnet/continuous-integration-workflow-template-for-net-core-desktop-apps-with-github-actions/>

Azure Key Vault

Azure Key Vault



Certificate



Password for the
certificate

Azure SignTool



[https://github.com/
vcsjones/AzureSignTool/](https://github.com/vcsjones/AzureSignTool/)

Azure DevOps



GitHub



Distribution

Store

Consumer apps

Store handles signing

Flights

Auto updates based on Store client



Non-Store

Non-Store Distribution (UNC and Web)

Requires trusted certificate

You are in control of update policy

Great for enterprises



Configuring automatic updates with the .appinstaller

```
<AppInstaller xmlns="http://schemas.microsoft.com/appx/appinstaller/2018"
  Uri="{AppInstallerUri}"
  Version="{Version}">
  <MainBundle Name="{Name}"
    Version="{Version}"
    Publisher="{Publisher}"
    Uri="{MainPackageUri}"/>
  <UpdateSettings>
    <OnLaunch HoursBetweenUpdateChecks="0"
      ShowPrompt="true"
      UpdateBlocksActivation="true" />
    <AutomaticBackgroundTask/>
    <ForceUpdateFromAnyVersion>true</ForceUpdateFromAnyVersion>
  </UpdateSettings>
</AppInstaller>
```

ms-appinstaller:?source=https://../..../app.appinstaller

Generating an .appinstaller file

- Automatically, as part of the Visual Studio build through the App Installer template
- On Azure DevOps, you can generate one using the MSIX Extensions

```
<?xml version="1.0" encoding="utf-8"?>
<AppInstaller Uri="{AppInstallerUri}"
              Version="{Version}"
              xmlns="http://schemas.microsoft.com/
                appx/appinstaller/2018">

  <MainBundle Name="{Name}"
              Version="{Version}"
              Publisher="{Publisher}"
              Uri="{MainPackageUri}" />

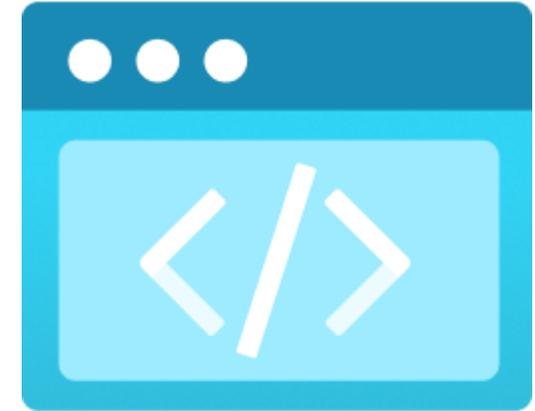
  <UpdateSettings>
    <OnLaunch HoursBetweenUpdateChecks="0"
              ShowPrompt="true"
              UpdateBlocksActivation="false" />
  </UpdateSettings>

</AppInstaller>
```

Where to deploy?



Azure Blob Storage
+ Static Websites



Azure Static Web Apps
(Preview)

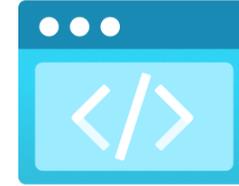


GitHub only for
the moment

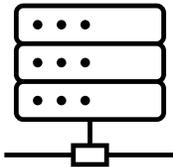
Where to deploy?



Azure Blob Storage
+ Static Websites



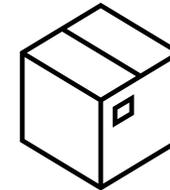
Azure Static Web Apps
(Preview)



Network share



GitHub Pages



Generic hosting via
FTP

Demo

The deployment pipeline

Container &
DevOps Day

@qmatteoq
matteo.pagani@microsoft.com

Slide e materiale su
<https://aspit.co/ContainerDevOpsDay-21>

Container &
DevOps Day