# Versioning, Packaging, and Distributing Class Libraries

**Jason Roberts**
.NET Developer

@robertsjason dontcodetired.com

# Overview

An overview of version numbering

- .NET assembly versions
- NuGet package versions
- Semantic versioning

Setting version numbers for class libraries

NuGet package creation

- Overview
- NuGet package properties
- Creating packages from VS and CLI
- Testing packages locally
- Setting package version numbers

An overview of automated package creation

Resources for further learning

# An Overview of Version Numbering

**.NET assembly version number (.dll file)**

**NuGet package version number**

# .NET Assembly Version Numbers

**Part of an assembly's "identity"**

- Assembly name
- Any culture information
- Strong naming public key

**"Strong-naming an assembly creates a unique identity for the assembly, and can prevent assembly conflicts."** [MS]

**"For .NET Core and .NET 5+, strong-named assemblies do not provide material benefits. The runtime never validates the strong-name signature, nor does it use the strong-name for assembly binding."** [MS]

**This version number is embedded in the assembly DLL file**

# .NET Assembly Version Numbers

| Major version number | Minor version number | Build number | Revision number |
| --- | --- | --- | --- |
| **2** | **1** | **5** | **4** |

# Major Version Number

**Changes (increments) with major new versions**

**Big new features added**

**Major rewrite**

**Indicates backward compatibility should not be assumed with previous version**

# Minor Version Number

**Changes (increments) with minor new versions**

**Smaller new features/enhancements**

**Indicates backward compatibility could be assumed with previous version**

# Build Number

**Changes when source code is recompiled**

**No changes/enhancements**

**Same source code**

**“Different build numbers might be used when the processor, platform, or compiler changes.”** [MS]

**Backwards compatibility can usually be assumed with previous version**

# Revision Number

**Changes for fixes**

- Bug fix
- Security vulnerabilty fixed

**No other changes/enhancements**

**“Assemblies with the same name, major, and minor version numbers but different revisions are intended to be fully interchangeable”** [MS]

“Subsequent versions of an assembly that differ only by build or revision numbers are considered to be Hotfix updates of the prior version.”

**Microsoft documentation**

https://docs.microsoft.com/en-us/dotnet/api/system.version?view=net-6.0

# .NET Core / .NET 5+

Assembly version numbers
don’t need to match exactly

# .NET Core / .NET 5+

Automatically load newer version numbers of the specified assembly name

# An Overview of Semantic Versioning

| Major version number | Minor version number | Patch version number | Optional labels |
| --- | --- | --- | --- |
| **2** | **1** | **5** | **-beta1** |

# Sematic Versioning (2.0.0)

**“SemVer” for short**

**Specific rules for version numbering**

**NuGet packages**

- NuGet 4.3.0+
- Visual Studio 2017 v15.3+

**Increment major version number when making any breaking changes to the API**

**Increment minor version number when adding backwards-compatible new functionality**

**Increment patch version number when making backwards-compatible bug fixes**

“...version numbers and the way they change convey meaning about the underlying code and what has been modified from one version to the next.”

**https://semver.org/**

# Sematic Versioning (2.0.0)

**Initial version**

- 1.0.0

**Fixed a bug**

- 1.0.1

**Fixed another bug**

- 1.0.2

**New backwards-compatible feature**

- 1.1.0 (patch reset to zero)

**Fixed a bug**

- 1.1.1

**New non backwards-compatible change**

- 2.0.0 (minor and patch reset to zero)

# Optional Labels

**Denote pre-release versions**

- Maybe be unstable / incompatible
- Add hyphen after patch number
- One or more dot-separated alphanumeric (and hyphen) strings
- 1.0.0-beta1

**Add build metadata**

- Add plus sign after patch or pre-release
- One or more dot-separated alphanumeric (and hyphen) strings
- 1.0.0+d241853866f20fc3e536cb3bca86c86c54b723ce

**1.0.0-beta2+36843**

Semantic versioning allows you to communicate changes in your class library project NuGet package to its consumers

# An Overview of NuGet Packages

**ZIP file that has a ".nupkg" extension**

**May contain one or many .NET assemblies**

- E.g. a class library project DLL

**May contain other files**

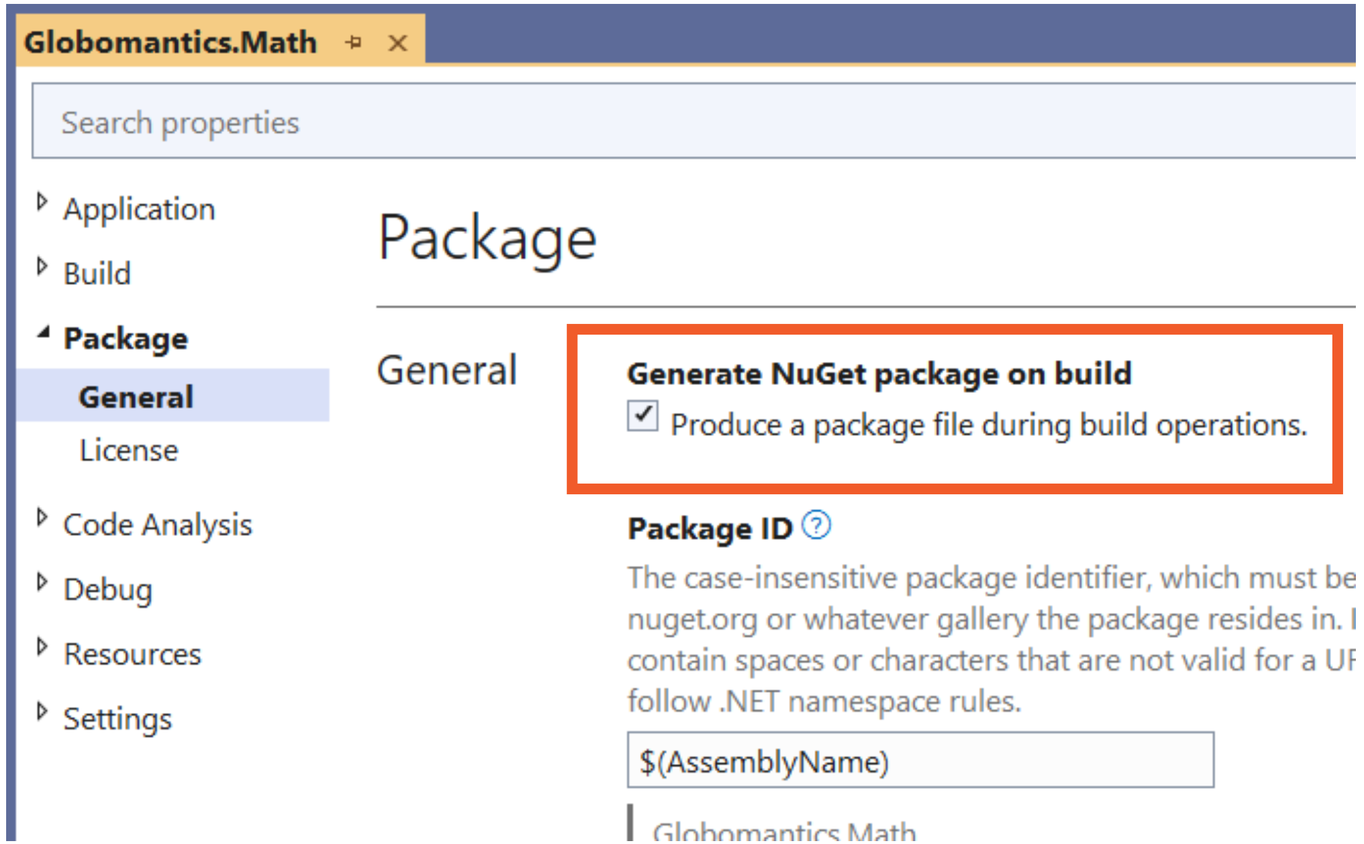
- Images
- Text files
- Powershell scripts, etc.

**Contains a manifest file**

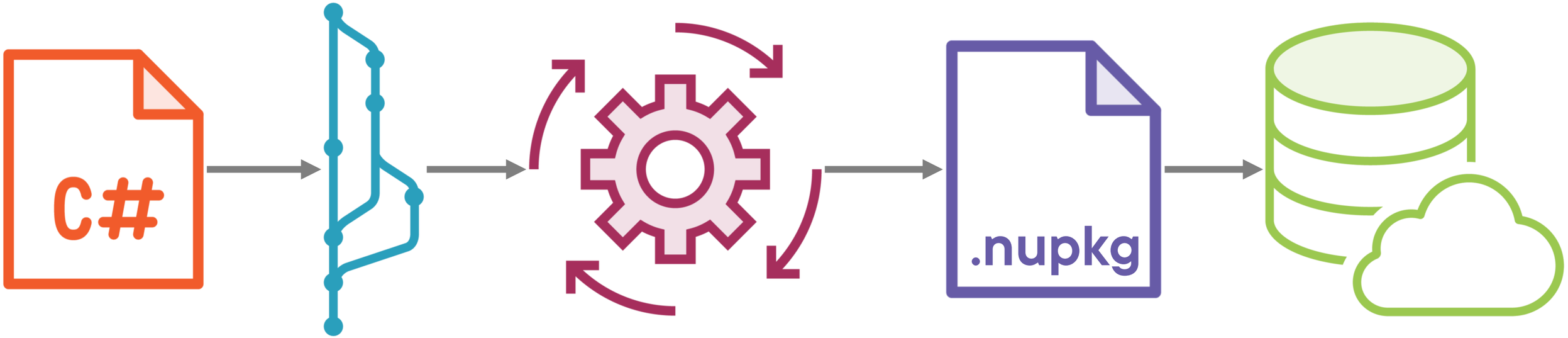
- Describes package contents
- Package version number (SemVer)
- Package ID, description, author, etc.

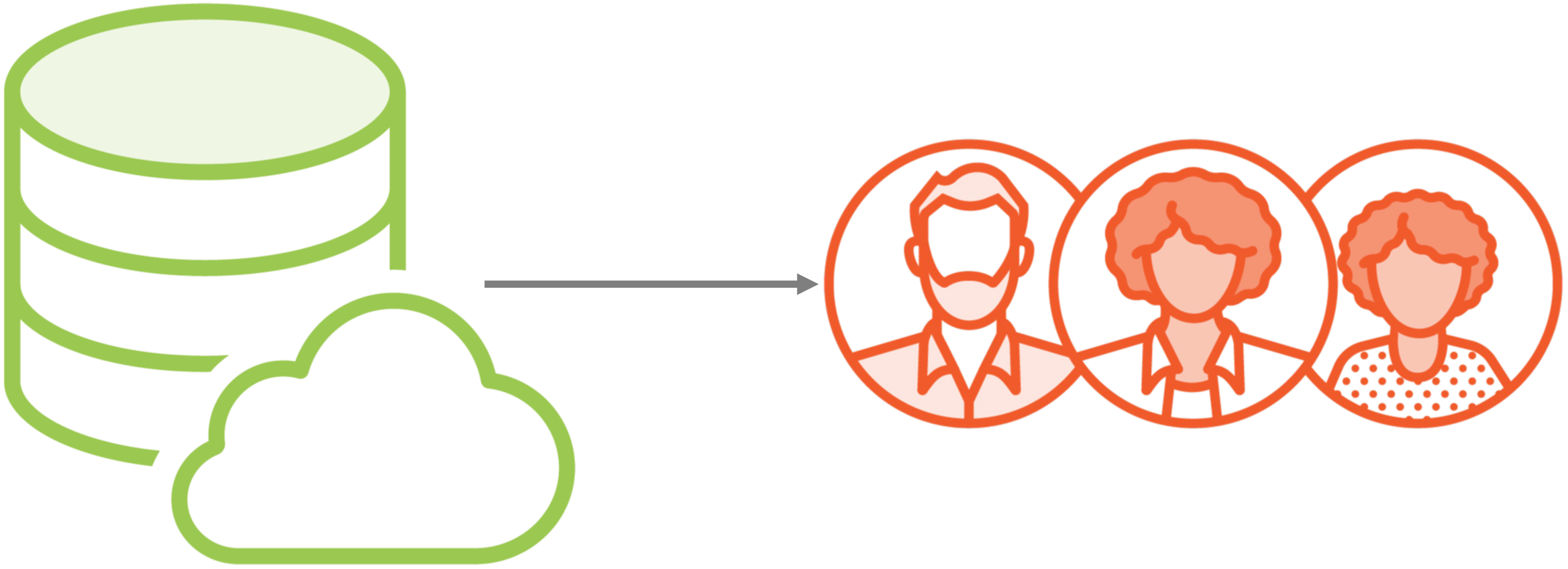
**Host on public (e.g. NuGet.org) or private host ("gallery")**

# Creating Packages Automatically



# Creating Packages Automatically





# Summary

.NET assemblies: major.minor.build.revision

NuGet packages: major.minor.patch (optional)

<AssemblyVersion>2.2.2.2</AssemblyVersion>

dotnet build /p:Version=4.0.0.1

Set NuGet package properties (e.g. description)

Create packages

- Pack in Visual Studio
- dotnet pack

Test packages locally in Visual Studio

<version> & dotnet pack /p:Version=2.0.0

An overview of automated package creation

# Resources and Further Learning

## Documentation and Courses

# C# Attributes

https://bit.ly/csharpattributes

Jason Roberts

# NUnit Testing

https://bit.ly/nunitcourse

Jason Roberts

# Working with Nulls in C#

https://bit.ly/psnull

Jason Roberts

# Documentation

**NuGet properties [MS]**

**https://bit.ly/psnuget**

**Multi-targeting guidance [MS]**

**https://bit.ly/psmulti**

**Target framework TFMs and preprocessor symbols [MS]**

**https://docs.microsoft.com/en-us/dotnet/standard/frameworks**

**Semantic Versioning 2.0**

**https://semver.org/spec/v2.0.0.html**

# My Pluralsight Course List

https://bit.ly/psjason

Jason Roberts