

+ 22

# C++ Dependencies Don't Have To Be Painful

## Why You Should Use a Package Manager

AUGUSTIN POPA



20  
22



# Welcome to CppCon 2022!

Join #visual\_studio channel on CppCon Discord  
<https://aka.ms/cppcon/discord>

Swing by the Microsoft booth today and tomorrow!

**Meet the Microsoft C++ team**

**Ask any questions**

**Discuss the latest announcements**

Take our survey

<https://aka.ms/cppcon>

# Which of these do you find frustrating about C++ development?

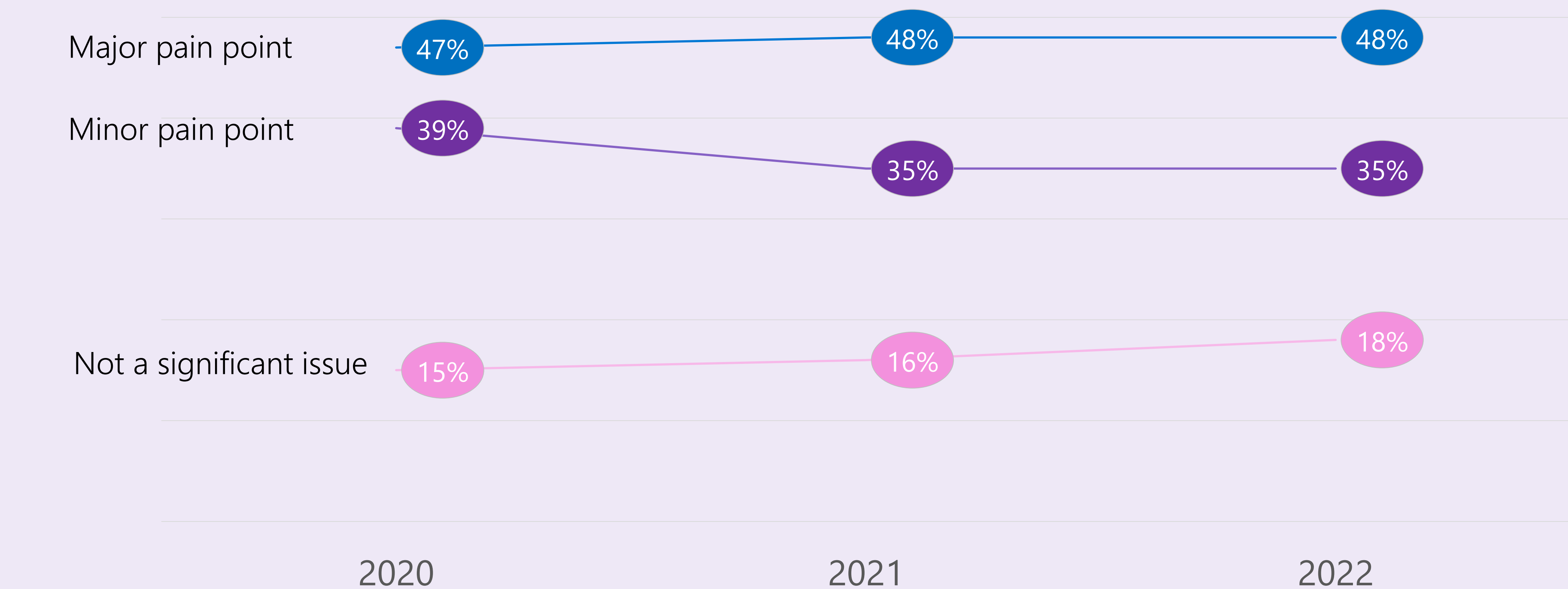
## Question from recent ISO C++ surveys

- Build times
- Managing CMake projects
- Debugging issues in my code
- Parallelism support
- Memory safety
- Managing Makefiles
- Managing MSBuild projects
- Setting up a CI pipeline from scratch
- Security issues
- Type safety
- Managing libraries my application depends on
- Moving existing code to the latest language standard
- Setting up a development environment from scratch

Source: ISO C++ surveys ([2020](#), [2021](#), [2022](#))

# Which of these do you find frustrating about C++ development?

Answer: Managing libraries my application depends on



Source: ISO C++ surveys ([2020](#), [2021](#), [2022](#))



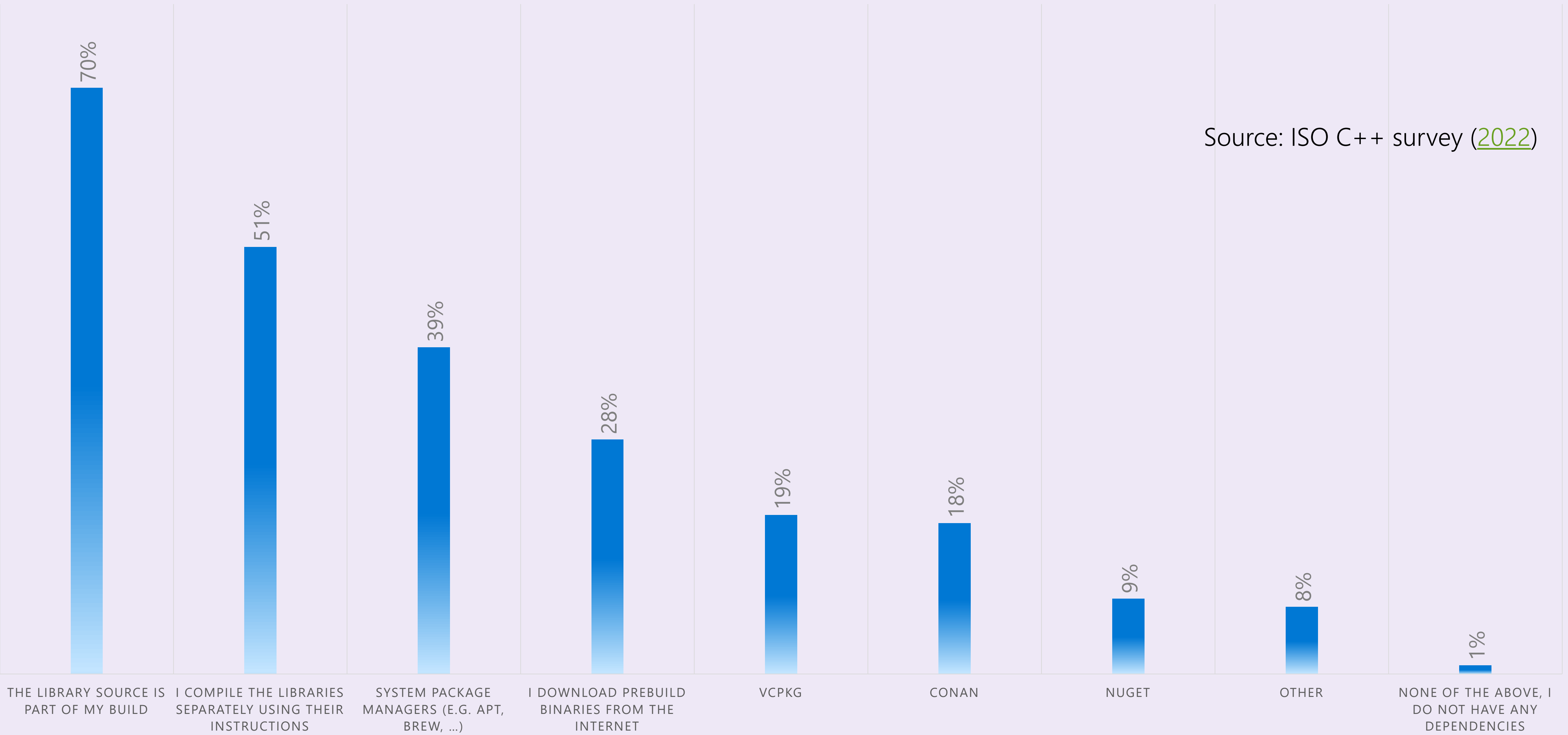
# How do you manage your 1<sup>st</sup> and 3<sup>rd</sup> party libraries? (check all that apply)

## Question from recent ISO C++ surveys

- The library source is part of my build
- I compile the libraries separately using their build instructions
- I download prebuild binaries from the Internet
- Other
- System package managers (apt, brew, ...)
- vcpkg
- Conan
- NuGet
- None of the above, I do not have any dependencies

Source: ISO C++ surveys ([2020](#), [2021](#), [2022](#))

# How do you manage your C++ 1<sup>st</sup> and 3<sup>rd</sup> party libraries? (Check all that apply)



# ABI compatibility between libraries and consuming project

A common C++ problem

uncommon in other programming languages

## How to break the ABI and your builds

Change the  
compiler

Change the  
compiler version

Change the  
target OS

Change the  
target  
architecture

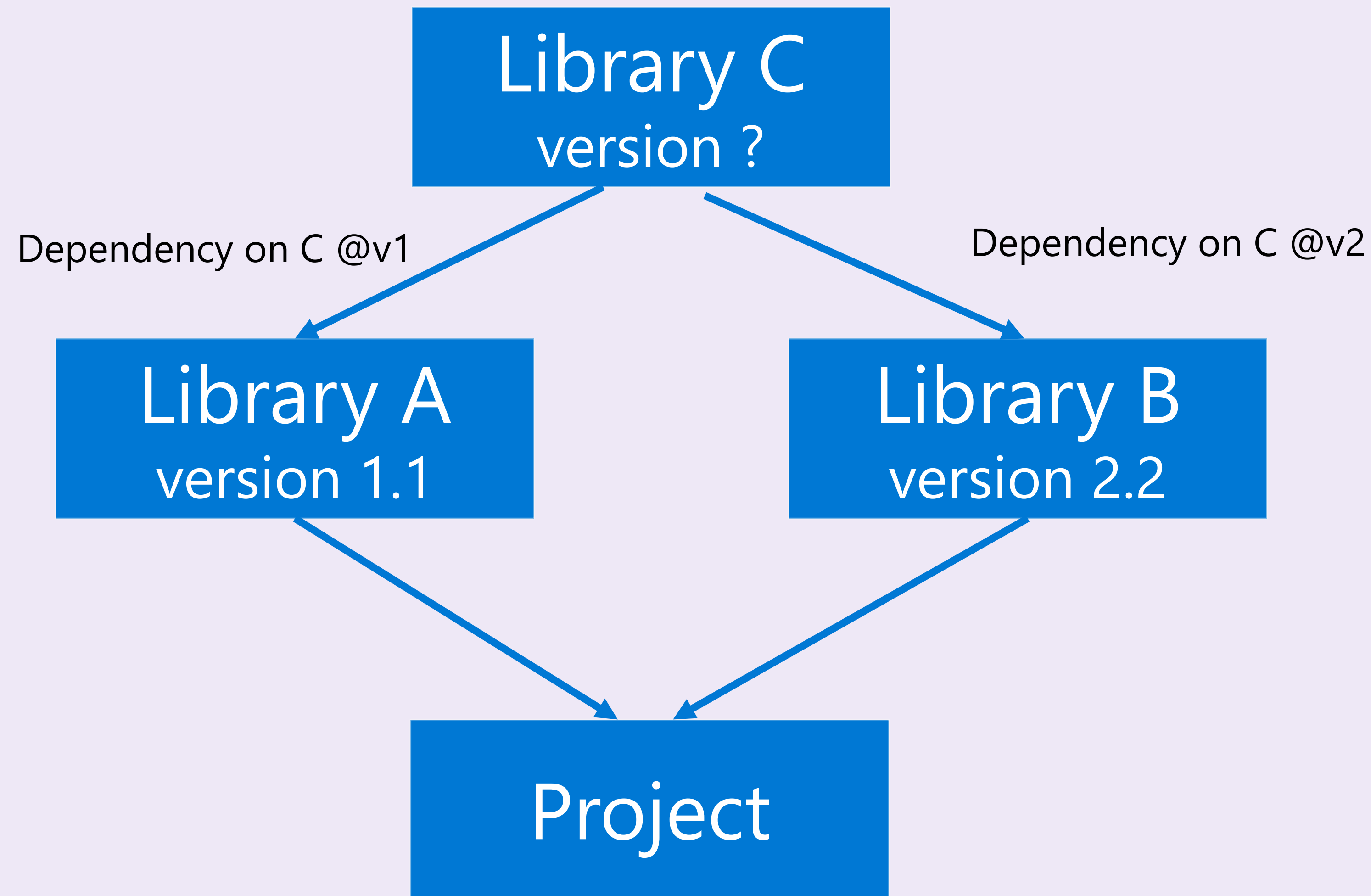
Add optional  
features to a  
library



# The “Diamond Problem”

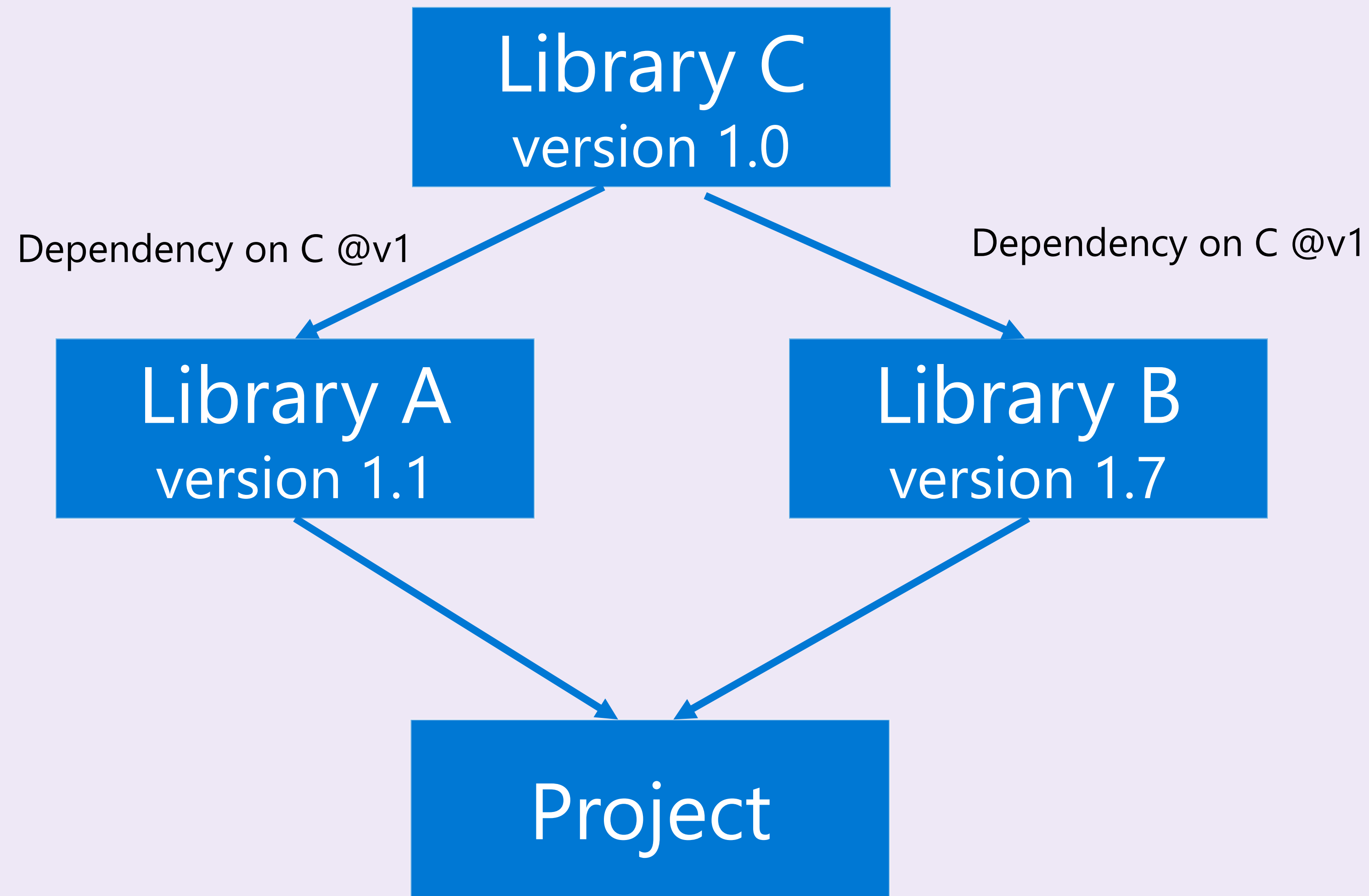
# The diamond problem of dependency management

## Example



# The diamond problem of dependency management

## Example Resolved



Package managers have  
adapted to solve some/all of  
these problems

## Baselines in vcpkg

### Making version matching an implementation detail

Catalog baseline: b60f003ccf5fe8613d029f49f835c8929a66eb61

boost@v1

openssl@v1

gtest@v1

...

Catalog baseline: a34d035cc48ye9699d350f49f835c3858a34bd20

boost@v2

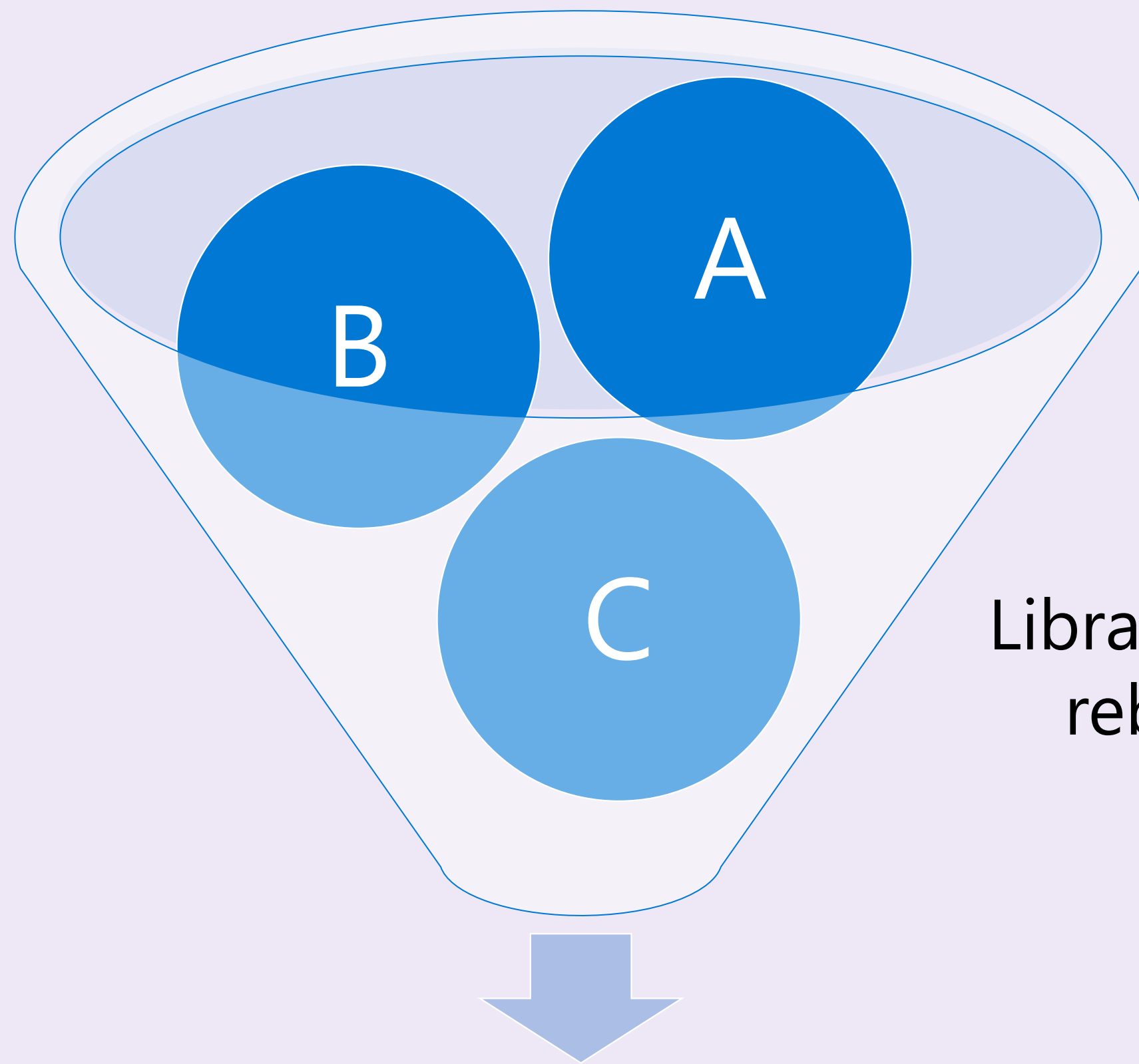
openssl@v1

gtest@v2

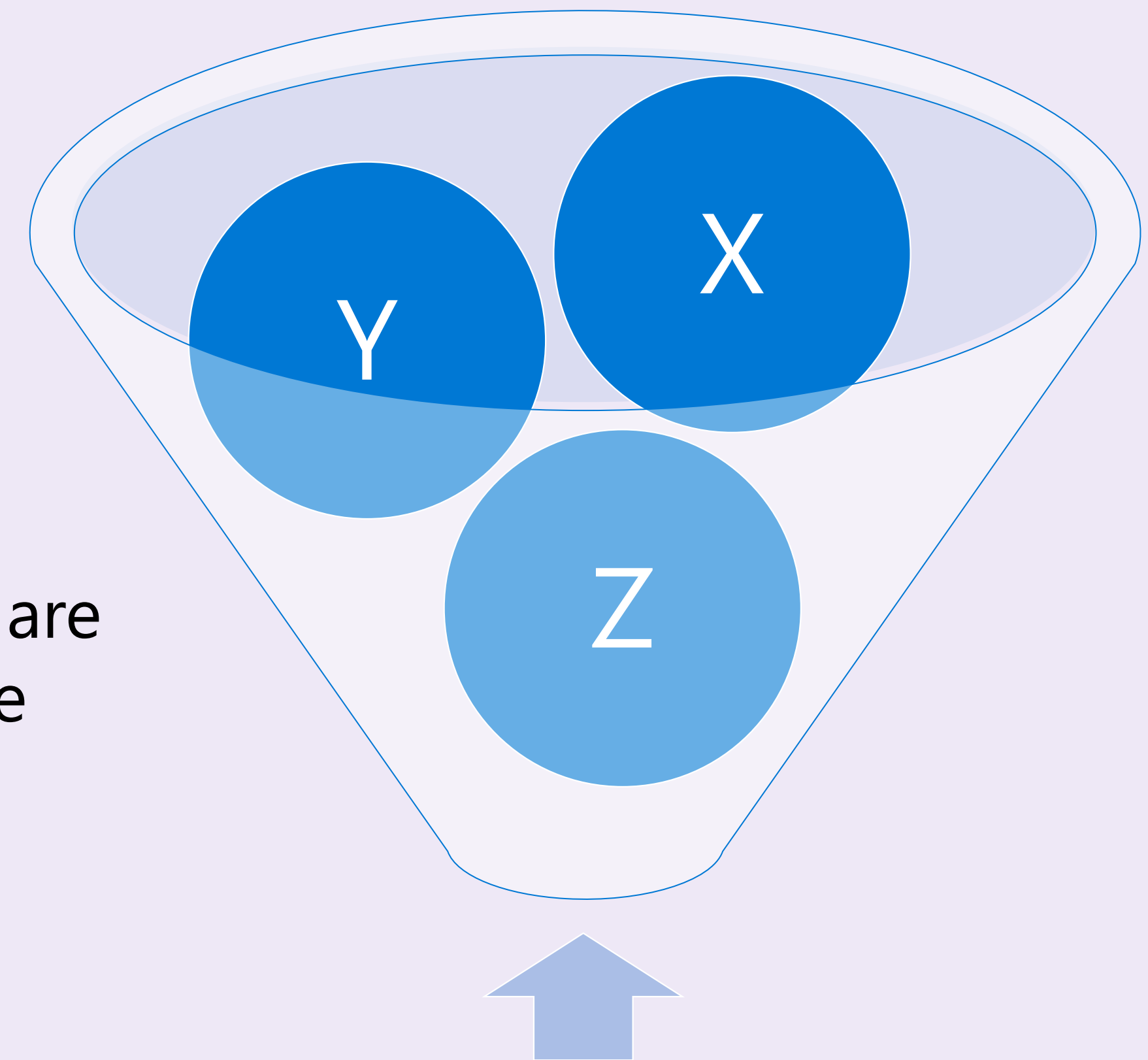
...

# The “cones of destruction” in vcpkg

## Preventing diamond problems



Library being updated uses A, B, C



Library being updated is used by X, Y, Z

Libraries A, B, C, X, Y, Z are  
rebuilt along with the  
changed library



# Keeping dependencies up to date is important

## Performance, new features, bug fixes, security, ...

- Many C++ developers may be hesitant to update their dependencies to a newer version due to the issues described earlier
- However, this results in very stale libraries
- Keeping your dependencies up-to-date is important for the long-term health of your codebase
- Using a package manager makes it easier to update your libraries over time

### OpenSSL Vulnerability | CISA

<https://www.cisa.gov/uscert/ics/advisories/ICSA-14-135-05> ▼

Apr 29, 2014 · the **openssl (heartbleed) vulnerability** was independently identified by both neel mehta of google security on april 1, 2014, and 2 days later by a team of security engineers riku, ...

### Multiple Vulnerabilities in OpenSSL Affecting Cisco ...

<https://tools.cisco.com/security/center/content/...> ▼

Sep 10, 2021 · On March 25, 2021, the **OpenSSL** Project released a security advisory, **OpenSSL Security Advisory [25 March 2021]**, that disclosed two **vulnerabilities**. Exploitation of these ...

### A Couple of Worrying Vulnerabilities Reported by OpenSSL

<https://vpnoverview.com/news/high-severity...> ▼

What Is Openssl?

Overview of The Open SSL V...

Important User Information

On August 24th, 2021 a security advisory report was released pertaining to **OpenSSL**. The release report details two **software vulnerabilities** within **OpenSSL**, one of which is marked as high-severity. The **vulnerabilities** can amount to a remote attacker acquiring full control of a vulnerable system, thus completely compromising it. The issues were repor...

[See more on vpnoverview.com](https://vpnoverview.com)

## Other neat C++ package manager benefits

- Support for building packages from source to overcome ABI compatibility issues
- Binary caches that can be downloaded from after a check that a rebuild isn't necessary
- Tight control over package versions (and easy version updates)
- Reproducible build environments (using manifests)
- Large, tested package catalogs
- Support for both open-source and closed-source packages
- Support for downloading packages from offline sources (for organizations disconnected from open Internet)
- Thousands of contributions from open-source contributors all over the world benefiting your development workflow (free engineering time 😊)

**Hypothesis: Developers are more likely to adopt a new process or workflow if they are more productive with it.**

# When you should consider a package manager

## Any 1 of these is enough

1. When your project has more than 1-2 dependencies, or you have dependencies of dependencies
2. When you have open-source dependencies
3. When your project has no dependencies, but you want to implement something that is already available in the public domain
4. When you are thinking about making your library header-only because it will make it more portable
5. If you are concerned about maintenance time or security

# What about modules?

## Won't they fix all our pain?

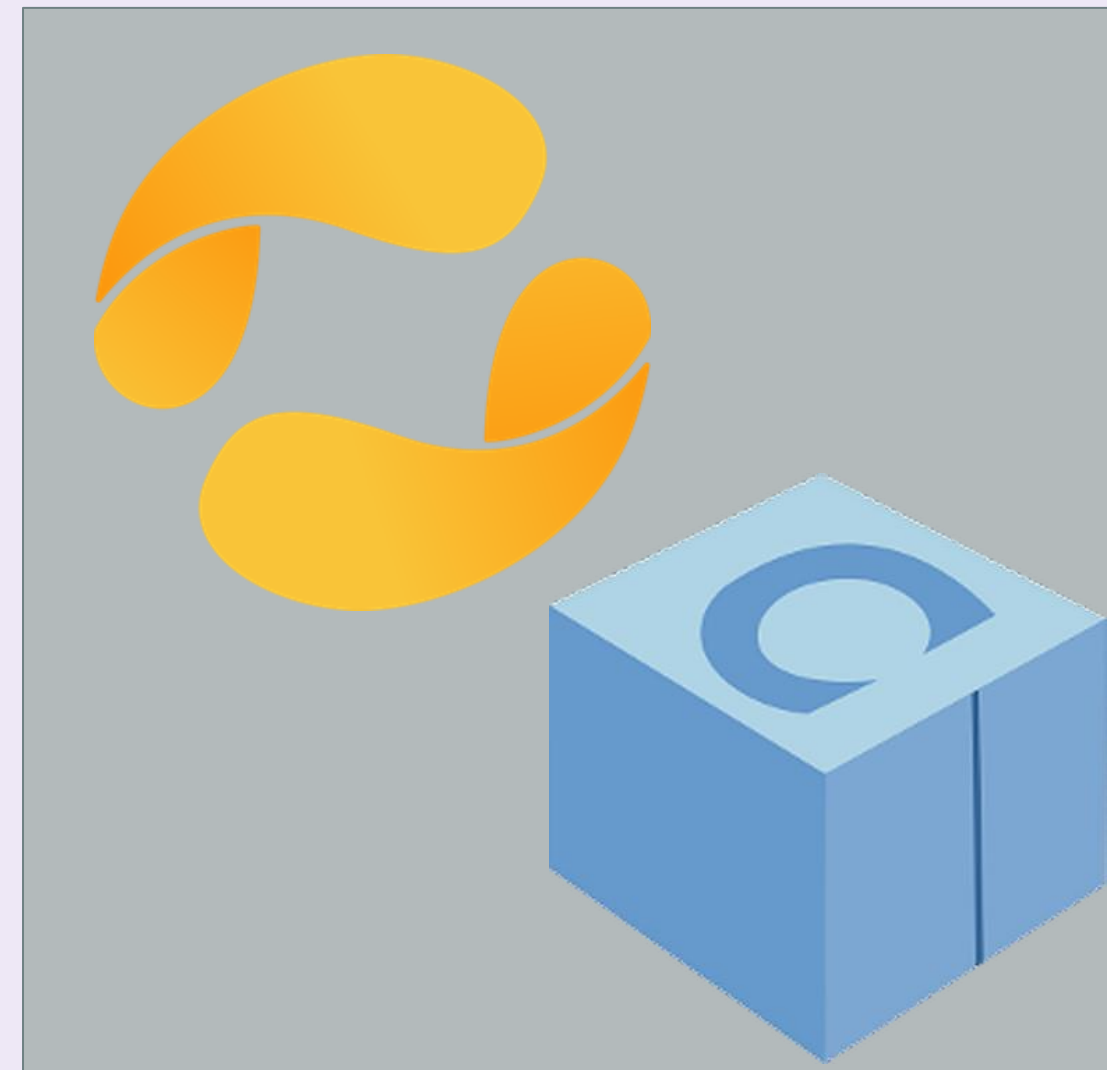
- Issues modules will address:
  - Over-use of headers during preprocessing (assuming you're not an expert with precompiled headers)
  - Macros and "using" declarations (by hiding them away from code outside the module)
  - Some ODR violations (separate translation units in separate modules)
  - Too many files in your repo (no need for source + headers as separate files)
  - Code architecture is clearer with separate logical components
- Issues modules won't fix:
  - Maintaining ABI stability within a dependency graph
  - Diamond problems
  - Migrating thousands of existing open-source libraries to modules (and even more closed-source ones)

# Types of package managers

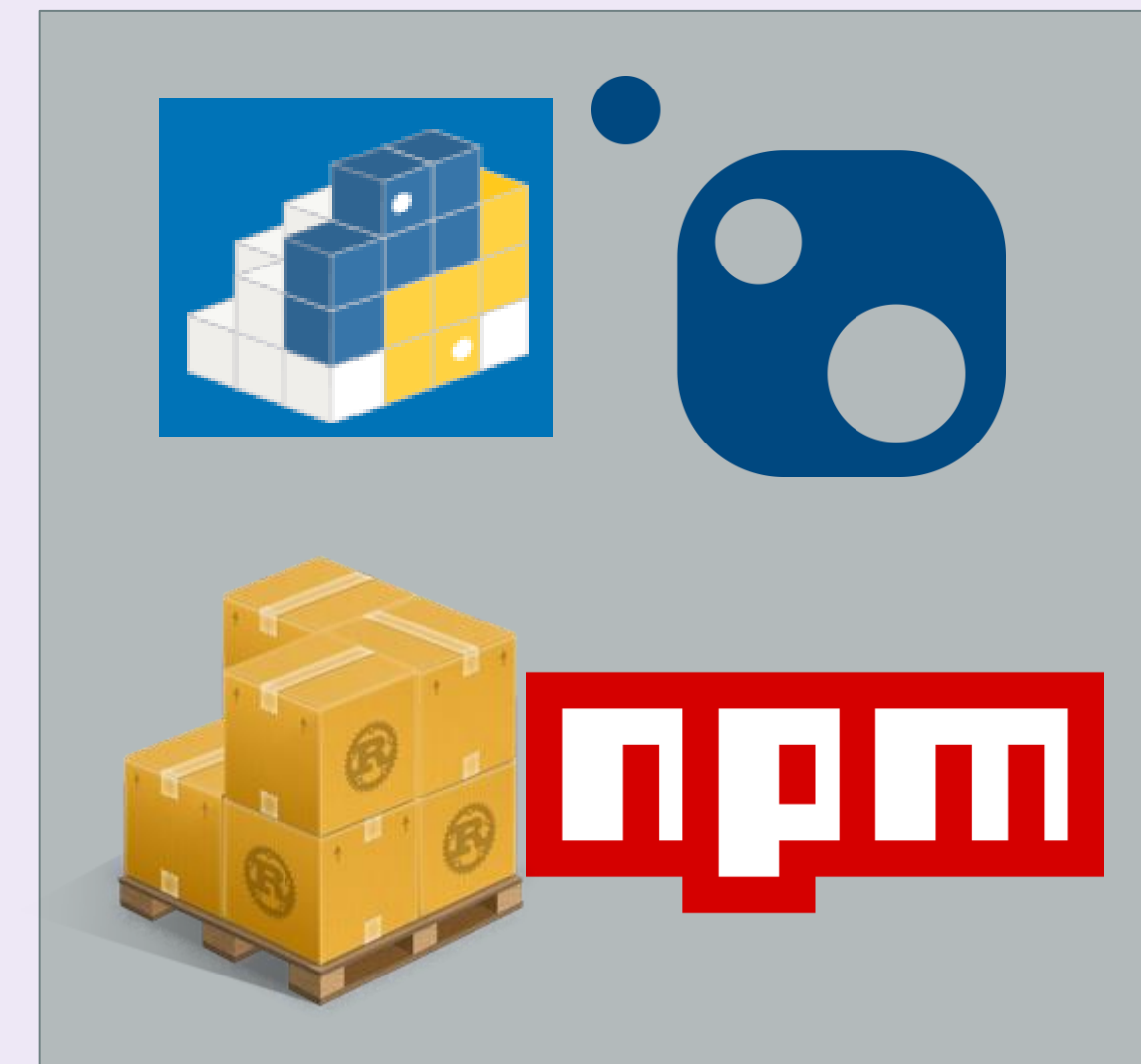
## As used by C++ developers



System package manager



C++ package manager



Language package manager  
(non-C++)



## System package managers

E.g. apt, yum, rpm, brew, winget, pacman, ...

- Designed for a particular operating system (OS) distribution
- Cross-platform projects must have special build logic for different OSes if system package managers are needed
- Packages are installed system-wide (usually elevated)
- Packages are typically acquired as-is, though some system package managers support build from source (e.g. Pacman)
- Not exclusive to C++ packages or even software development; can install apps, tools, and libraries for any workflow
- Don't typically provide first-class integration with build systems – however, since install paths are known defaults, your build system may find packages anyway
- Most popular on Linux

## C++ package managers

E.g. vcpkg, Conan

- Tailored for C++ development with more advanced features
- Have ways to address diamond dependency problems
- Support building packages from source or downloading valid prebuilt binaries
- Support a large variety of open-source libraries out-of-the-box
- Also support private libraries
- Support acquisition of build tools, platform SDKs, debuggers, and other tools needed for a working C++ environment for cross-platform development
- Work across multiple platforms, architectures, and compilers

## Non-C++ package managers

E.g. NuGet, npm, Cargo, pip

- A single-language package manager being repurposed for use with other programming languages as well
- Useful in limited scenarios (e.g. when a developer primarily uses another programming language than C++, and doesn't want a new package acquisition workflow for C++)
- NuGet (a .NET package manager) is the most common example used by C++ developers (9% in 2022 ISO C++ survey).
- Scenarios NuGet does not address well:
  - ABI violations/diamond problems: no support for building from source, for different compilers, compiler versions target architectures, target OS. Need a separate package for each configuration.
  - Build systems that are not MSBuild
  - Non-Windows operating systems (while technically possible under Mono or dotnet CLI, it's still not first-class support for C++)
- Since non-C++ language package managers do not address unique C++ requirements, not recommended for C++ except for developers touching C++ assets that have no plans to ever modify them

# Which type of package manager should you use for C++ packages?

## There is a time and place for everything

- **If** you work in or target primarily one system, and you do not update your dependencies frequently use a system package manager
- **Else If** you work in primarily another programming language, use that language's primary package manager
- **If** you need a system-specific asset (e.g. a Linux-only graphics library for your video game's Linux port), and that package is not easily available in a C++ language package manager, use a system package manager
- **Else**, use a C++ language package manager, which helps you resolve ABI issues, diamond problems, offers you access to much wider variety of C++ packages (and updated package versions). Also great for installing per-project dependencies so that other projects can have separate versions of the same dependency.

# Reproducible development environments

## Getting compilers, debuggers, build systems, graphical assets, ...

- Package managers can do more than just get library dependencies
- C++ developers need dev tools as well (build tools, debuggers, platform SDKs, static analysis tools, runtime analysis tools, build systems, ...)
- System package managers have always offered a wide variety of packages (including dev tools)
- More recently, C++ language package managers are too (including vcpkg and Conan)
- It is important to be able to bootstrap a C++ development environment in an automated and reproducible way
- This ensures consistency between different dev machines and local dev environments and CI
- Use manifests to declare devtool and library dependencies

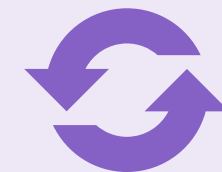


# How to manage dependencies well

## Principles to keep in mind



Be able to build dependencies from source (when necessary)



Keep your dependencies up-to-date



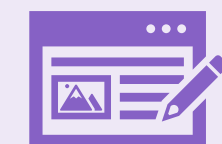
Cross-platform should be first class experience



Make your build environment reproducible



Do download prebuilt binaries, if they are verified



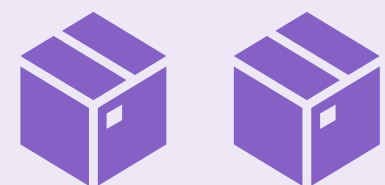
Simplify workflow for authoring and publishing dependencies



Take advantage of existing open-source solutions



Enforce ABI requirements across all packages, not one at a time



Use more than one package manager if it improves your productivity



# C++ package managers must be accessible and productive for ALL C++ developers

Great strides have been made over the past several years,  
but we can do so much more...

# Consider contributing to an open-source package manager

Rather than maintaining an in-house solution

# Microsoft sessions at CppCon

Monday 12th

- **GitHub Features Every C++ Developer Should Know** – Michael Price
- **The Imperatives Must Go** – Victor Ciura
- **What's New in C++ 23** – Sy Brand
- **C++ Dependencies Don't Have to Be Painful** – Augustin Popa
- **How Microsoft Uses C++ to Deliver Office** – Zachary Henkel

Tuesday 13th

- **High-performance Load-time Implementation Selection** – Joe Bialek, Pranav Kant
- **C++ MythBusters** – Victor Ciura

Wednesday 14th

- **memory-safe C++** - Jim Radigan

Thursday 15th

- **What's New for You in Visual Studio Code** – Marian Luparu, Sinem Akinci
- **Overcoming Embedded Development Tooling Challenges** – Marc Goodner
- **Reproducible Developer Environments** – Michael Price

Friday 16th

- **What's New in Visual Studio 2022** – Marian Luparu, Sy Brand
- **C++ Complexity (Keynote)** – Herb Sutter

I would love to hear about how  
you manage dependencies and  
what you are struggling with.

# Enjoy the rest of the conference!

Join #visual\_studio channel on CppCon Discord  
<https://aka.ms/cppcon/discord>

Swing by the Microsoft booth today and tomorrow!

**Meet the Microsoft C++ team**

**Ask any questions**

**Discuss the latest announcements**

Take our survey

<https://aka.ms/cppcon>