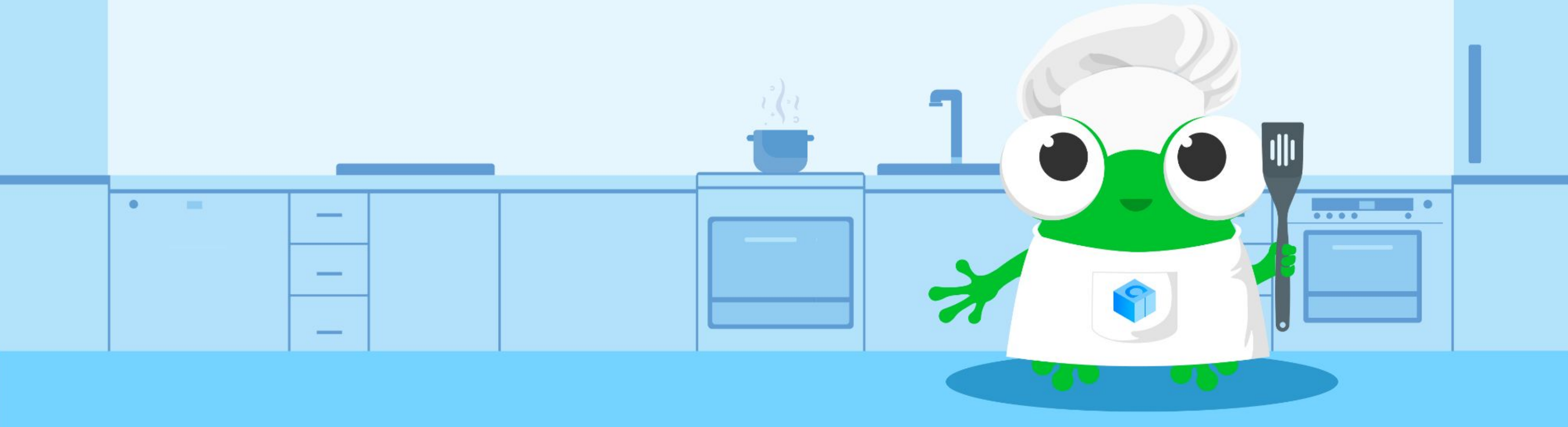
WHAT'S NEW IN CONAN 2.0

The lessons we have learned from the C++ ecosystem

Christopher McArthur, Conan Developer Advocate



JFrog

CONAN
C/C++ Package Manager

What is Conan?

C and C++ Package Manager

What is the role of a package manager?

- Easily install dependencies
  - `conan install --requires=spdlog/1.11.0`

How is Conan different?

- Enable you to build and distribute binaries

We offer **JFrog's ConanCenter** with 1500+ open-source projects with over 100 configuration (os, compiler, arch) being created and publish to help accelerate OSS.

- Open-Source
- Distributed
- Scalable and flexible
- Remotes + Server

Supports

- CMake, Meson, Autotool, etc…
- Any Platforms
- More than just CMakeLists.txt

JFrog | CONAN C/C++ Package Manager

Why Conan?

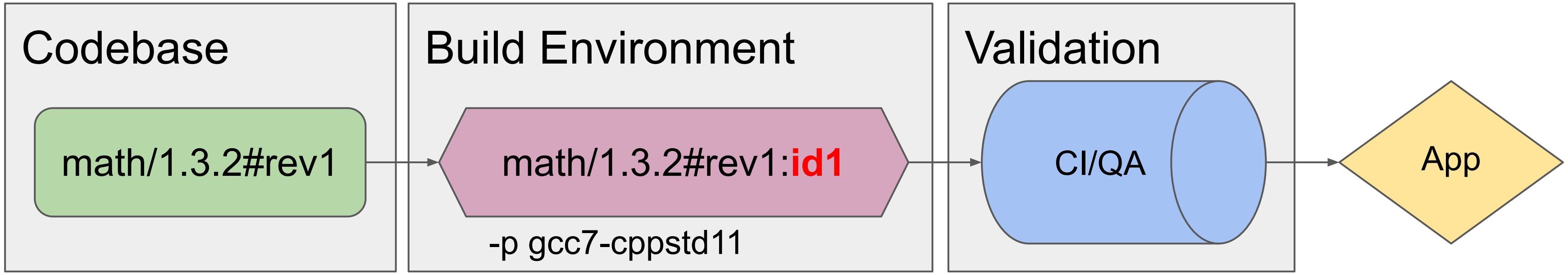
- Key difference between Conan and other C/C++ package manager is the focus on binaries. Being able to create packages that can be **re-used** across multiple teams throughout an organization. Enables flexibility and scalability. Framework for doing DevOps and Package Management in a very enterprise ready manner.

You can model **platform configurations** and the **linkage between libraries**. To deterministically know what to build but more importantly what you already have to save time and money!

A common problem

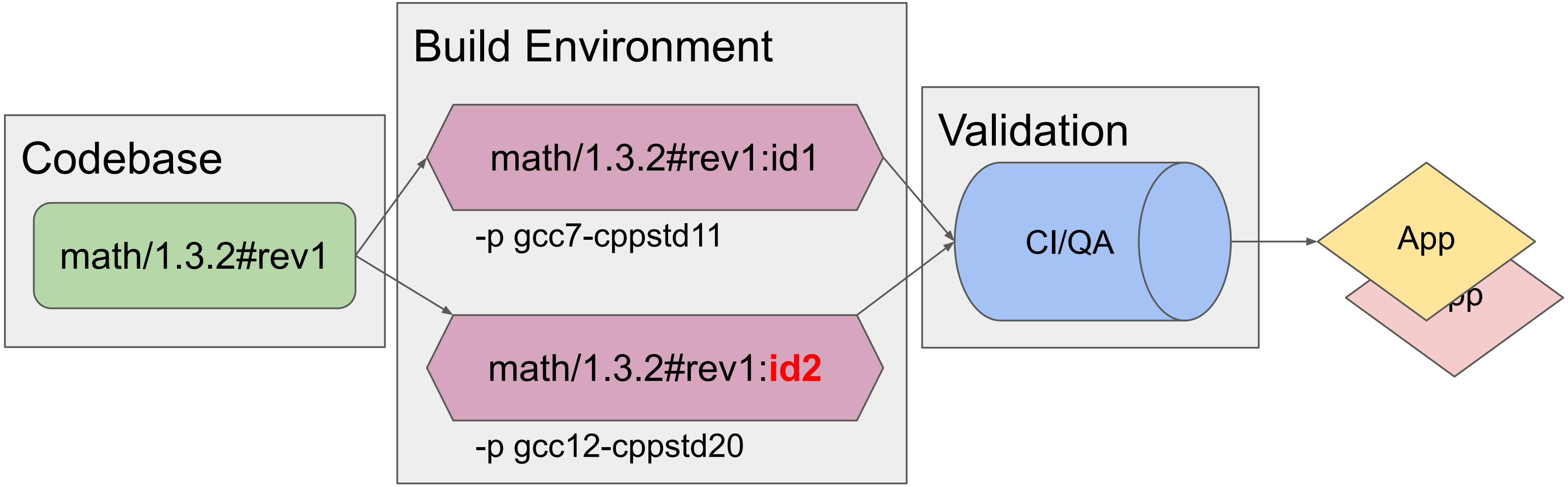
- Most C++ developers are still runnings C++11- but in a perfect world who doesn't want to update?

Today (build and shipped to customer)



A common solution

Future (Current development being validated along side LTS)



A solve problem?

You’re probably wonder why this sounds familiar?

- Probably doing this with Debug/Release
- Maybe Windows/Linux separation?

Starting into the “DevOps” is knowing what packages need to built, tracking existing binary across the software development life cycle.

JFrog CONAN C/C++ Package Manager

What’s new in Conan 2.0

Everything is new!

5 years, without breaking

60% new code, 20% backports

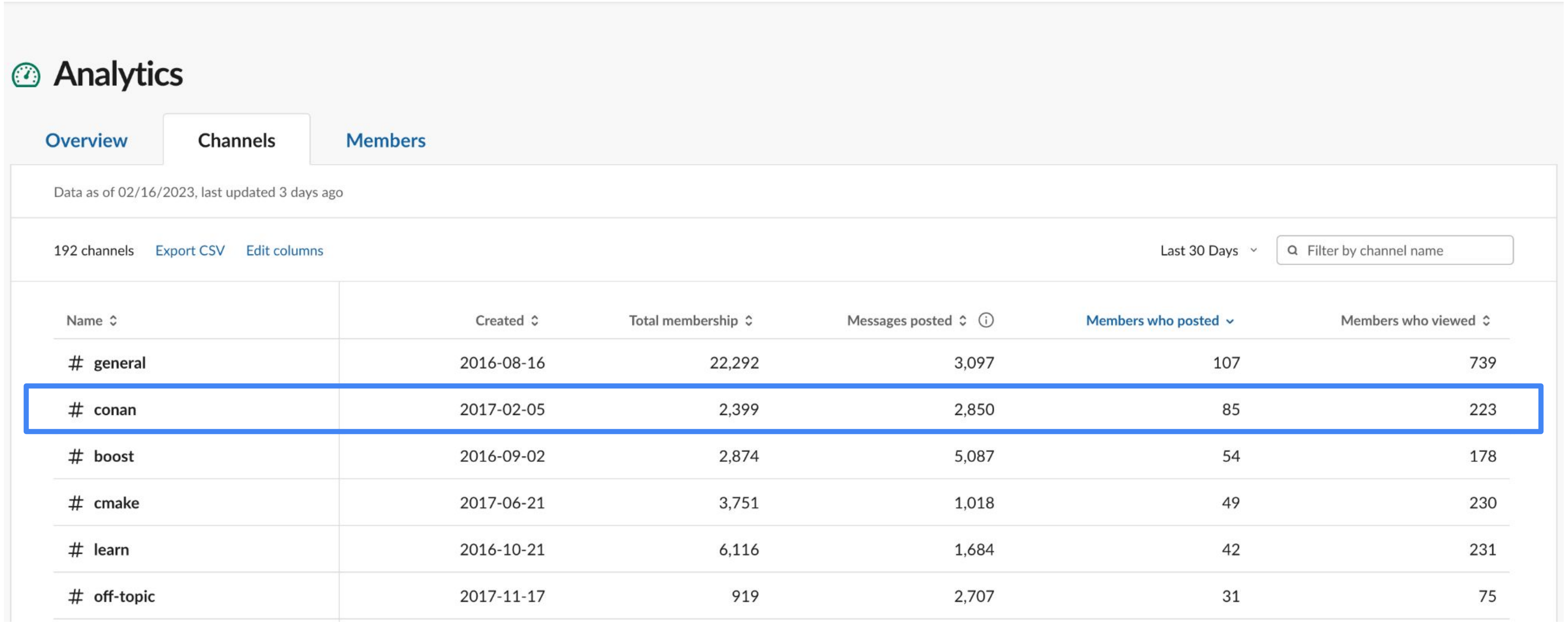
1.X ⇔ 2.0 compatible syntax subset

1.0

2.0

We have been listening to you.

CppLang #conan slack



Analytics

Overview | Channels | Members

Data as of 02/16/2023, last updated 3 days ago

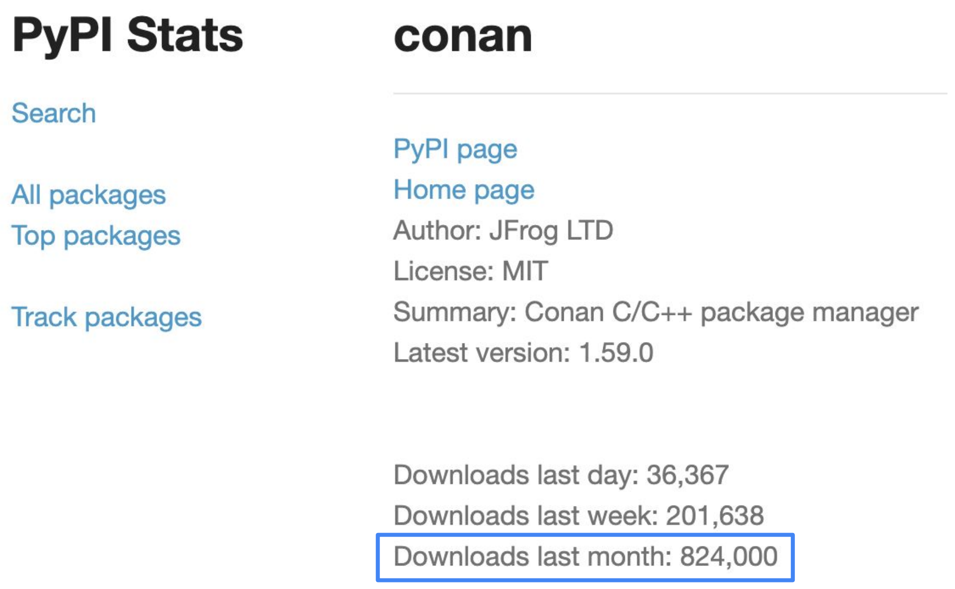
192 channels Export CSV Edit columns Last 30 Days Filter by channel name

Name	Created	Total membership	Messages posted	Members who posted	Members who viewed
# general	2016-08-16	22,292	3,097	107	739
# conan	2017-02-05	2,399	2,850	85	223
# boost	2016-09-02	2,874	5,087	54	178
# cmake	2017-06-21	3,751	1,018	49	230
# learn	2016-10-21	6,116	1,684	42	231
# off-topic	2017-11-17	919	2,707	31	75

JFrog | CONAN C/C++ Package Manager

PyPI downloads (Conan tool)

- 800K downloads/month from PyPI
- Designated as PyPI critical project (1% of most downloaded in whole PyPI)



PyPI Stats

Search

All packages
Top packages

Track packages

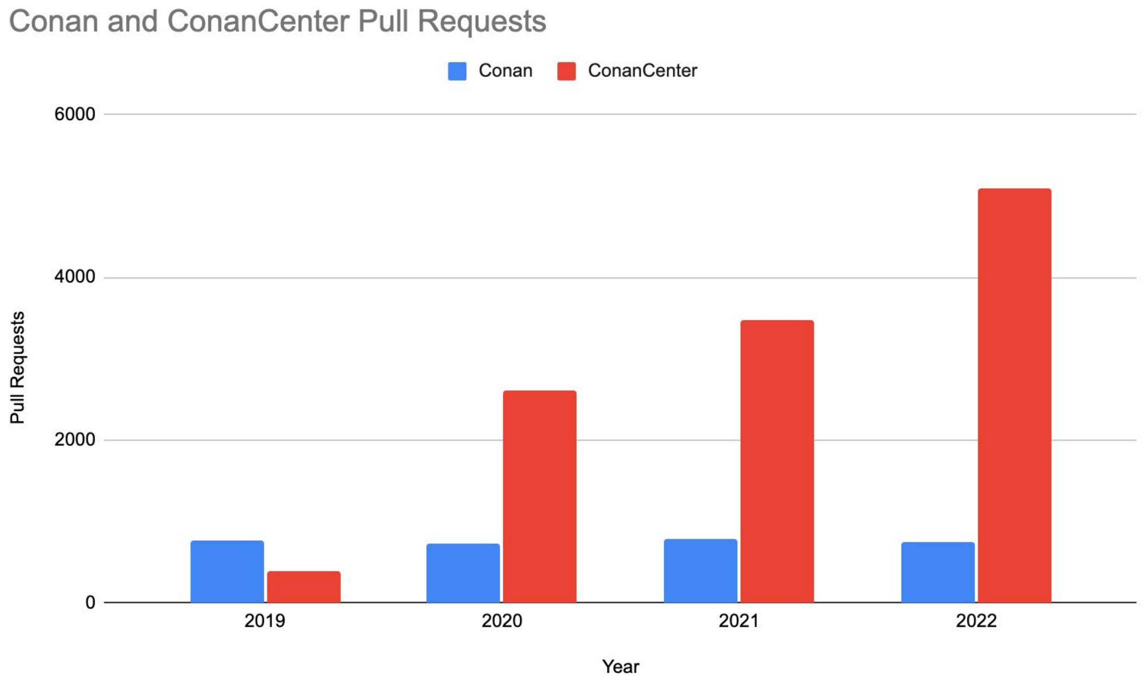
conan

PyPI page
Home page
Author: JFrog LTD
License: MIT
Summary: Conan C/C++ package manager
Latest version: 1.59.0

Downloads last day: 36,367
Downloads last week: 201,638
Downloads last month: 824,000

JFrog | CONAN C/C++ Package Manager

Github PRs

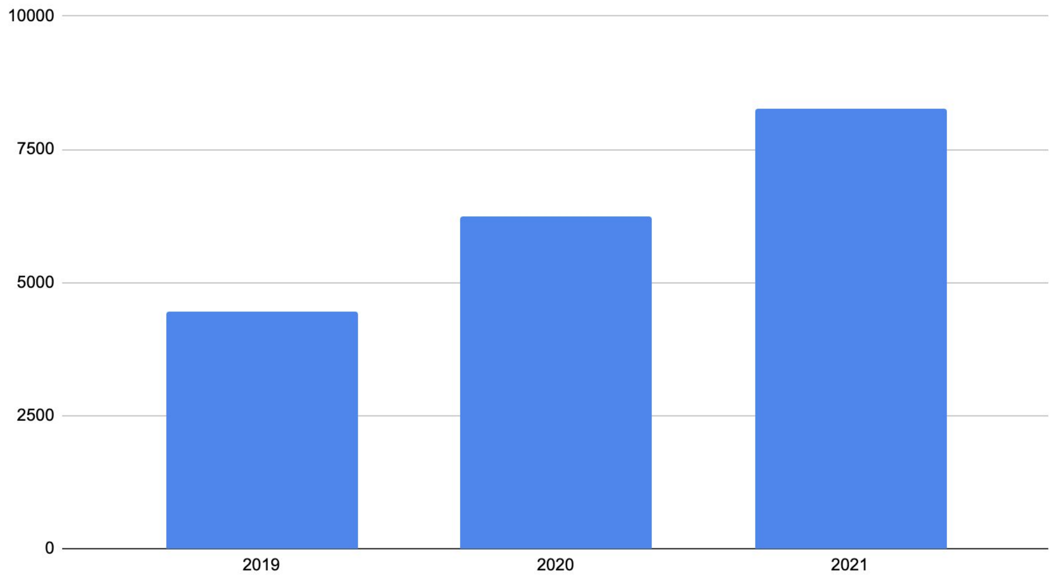

Conan and ConanCenter Pull Requests
Conan
ConanCenter
6000
4000
2000
0
Pull Requests
2019
2020
2021
2022
Year

Support

+2000 Github issues / year

100 hr/year user video calls

Direct support (slack, almost daily)

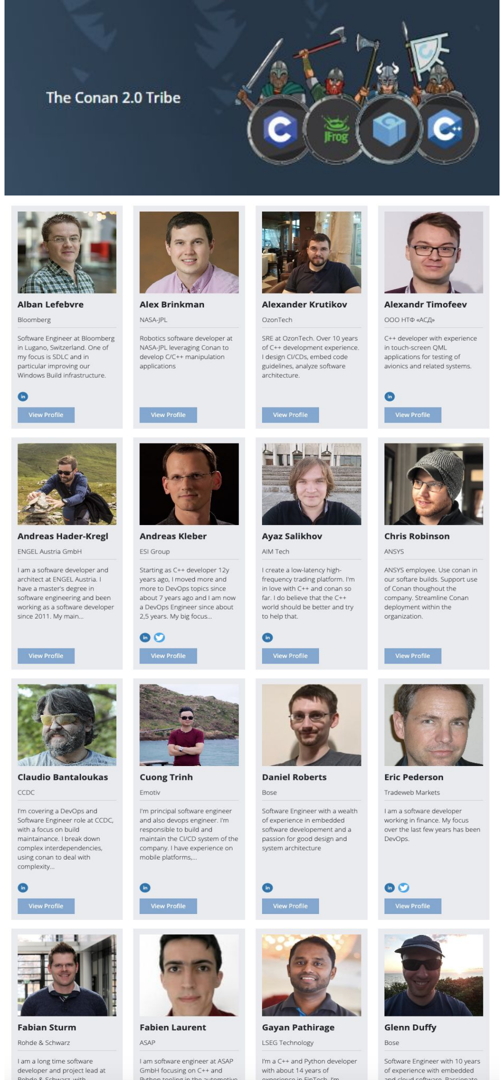


Artifactory servers running Conan in production and telemetry enabled (no firewalls)

JFrog | CONAN C/C++ Package Manager

Tribe 2.0 (conan.io/tribe.html)

Bose	ASAP
TomTom	Rti
Continental	Zeiss
Nasa	Nasdaq
Apple	Plex
Ansys	Keysight
Bloomberg	Datalogics
Rohde & Schwarz	VMWare
Bosch	… 50 more



Alban Lefebvre
Bloomberg
Software Engineer at Bloomberg in Lugano, Switzerland. One of my focus is SDLC and in particular improving our Windows Build infrastructure.
View Profile

Alex Brinkman
NASA-JPL
Robotics software developer at NASA-JPL leveraging Conan to develop C/C++ manipulation applications
View Profile

Alexander Krutikov
OzonTech
SRE at OzonTech. Over 10 years of C++ development experience. I design CI/CDs, embed code guidelines, analyze software architecture.
View Profile

Alexandr Timofeev
ООО НТФ «АСД»
C++ developer with experience in touch-screen QML applications for testing of avionics and related systems.
View Profile

Andreas Hader-Kregl
ENGEL Austria GmbH
I am a software developer and architect at ENGEL Austria. I have a master's degree in software engineering and been working as a software developer since 2011. My main...
View Profile

Andreas Kleber
ESI Group
Starting as C++ developer 12y years ago, I moved more and more to DevOps topics since about 7 years ago and I am now a DevOps Engineer since about 2,5 years. My big focus...
View Profile

Ayaz Salikhov
AIM Tech
I create a low-latency high-frequency trading platform. I'm in love with C++ and conan so far. I do believe that the C++ world should be better and try to help that.
View Profile

Chris Robinson
ANSYS
ANSYS employee. Use conan in our softare builds. Support use of Conan thoughout the company. Streamline Conan deployment within the organization.
View Profile

Claudio Bantaloukas
CCDC
I'm covering a DevOps and Software Engineer role at CCDC, with a focus on build maintainance. I break down complex interdependencies, using conan to deal with complexity...
View Profile

Cuong Trinh
Emotiv
I'm principal software engineer and also devops engineer. I'm responsible to build and maintain the CI/CD system of the company. I have experience on mobile platforms,...
View Profile

Daniel Roberts
Bose
Software Engineer with a wealth of experience in embedded software developement and a passion for good design and system architecture
View Profile

Eric Pederson
Tradeweb Markets
I am a software developer working in finance. My focus over the last few years has been DevOps.
View Profile

Fabian Sturm
Rohde & Schwarz
I am a long time software developer and project lead at Rohde & Schwarz, with

Fabien Laurent
ASAP
I am software engineer at ASAP GmbH focusing on C++ and Python tooling in the automotive

Gayan Pathirage
LSEG Technology
I'm a C++ and Python developer with about 14 years of experience in FinTech. I'm

Glenn Duffy
Bose
Software Engineer with 10 years of experience with embedded and cloud software. Passionate

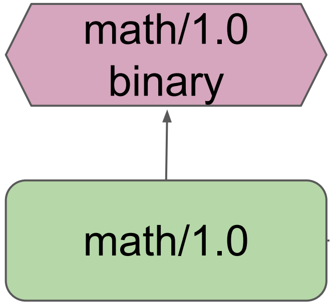
Overview

- 3 lessons:
  - Learning to fly
  - Repeating yourself
  - Build a dam
- Conclusions

JFrog | CONAN C/C++ Package Manager

1. Learning to fly

Conanfile: A package "recipe"



```
$ git clone … math && cd math
$ conan create .
```

math/conanfile.py

```python
from conan import ConanFile

class Math(ConanFile):
    name = "math"
    version = "1.0"

    def source(self): ...
    def build(self): ...
    def package(self): ...
```

Conan 1.X dependency model: Transitive deps

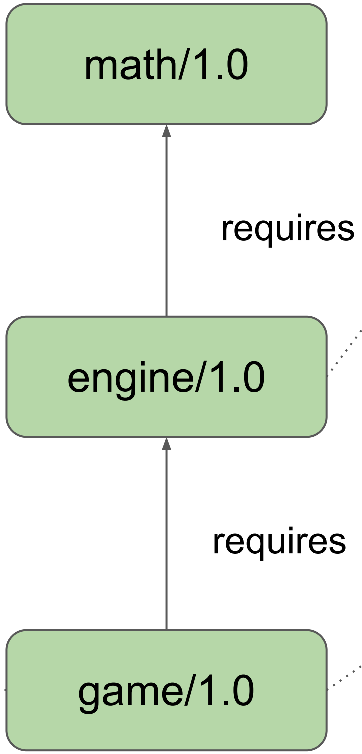
math-config.cmake

```
set_property(TARGET math::math …)
```

engine-config.cmake

```
set_property(TARGET engine::engine …)
```

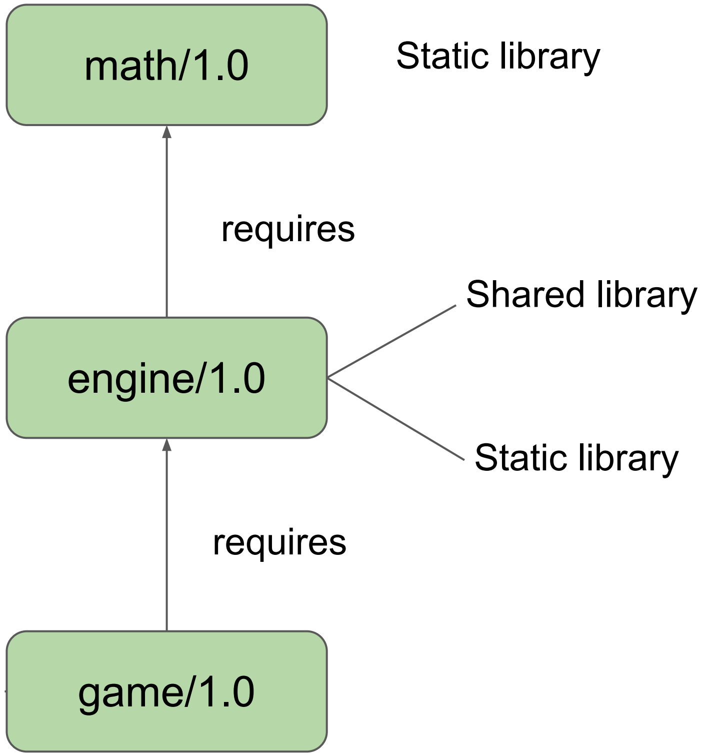
```
$ git clone … game && cd game
$ conan install .
$ cmake …
```



```
from conan import ConanFile

class Engine(ConanFile):
    requires = "math/1.0"
```

```
from conan import ConanFile

class Game(ConanFile):
    requires = "engine/1.0"
    generators = "CMakeDeps"
```

Conan 1.X dependency model: Transitive deps



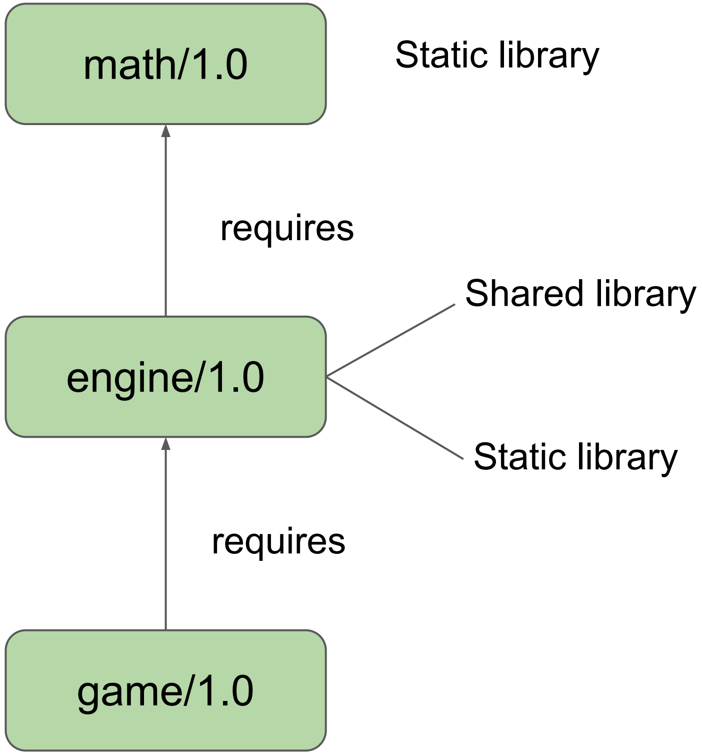
math-config.cmake

```
set_property(TARGET math::math ...)
```

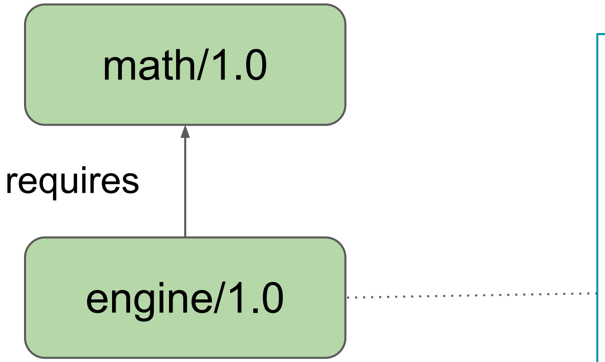
engine-config.cmake

```
set_property(TARGET engine::engine ...)
```

Learning to fly



Conan 2.0 proposal

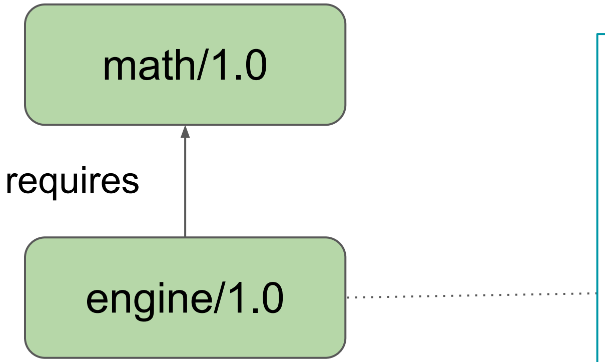


engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0")
```

JFrog CONAN C/C++ Package Manager

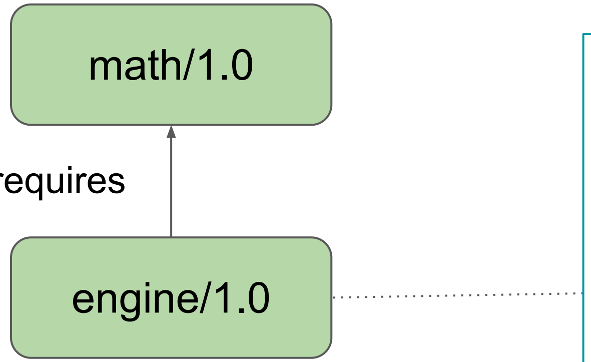
Conan 2.0 proposal: Requirement traits



engine/conanfile.py

```
from conan import ConanFile

class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True)
```

Conan 2.0 proposal: Requirement traits



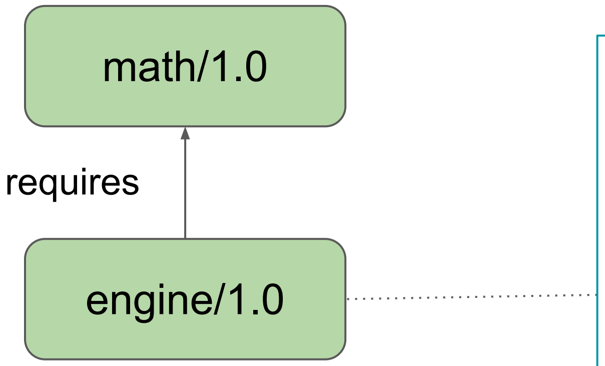
engine/conanfile.py

```
from conan import ConanFile


class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=True)
```

math-config.cmake

```
set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)
set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)
```

Conan 2.0 proposal: Requirement traits



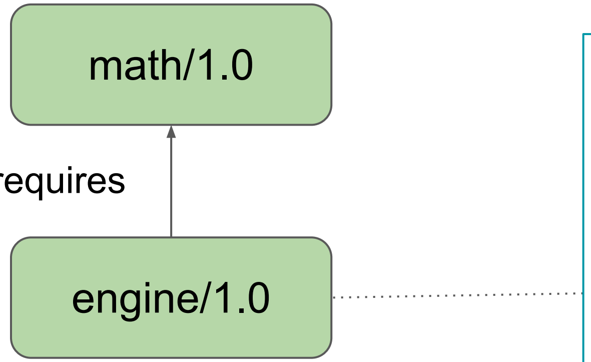
engine/conanfile.py

```
from conan import ConanFile


class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=False, libs=True)
```

math-config.cmake

```
set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)
~~set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES~~ ...)
```

Conan 2.0 proposal: Requirement traits



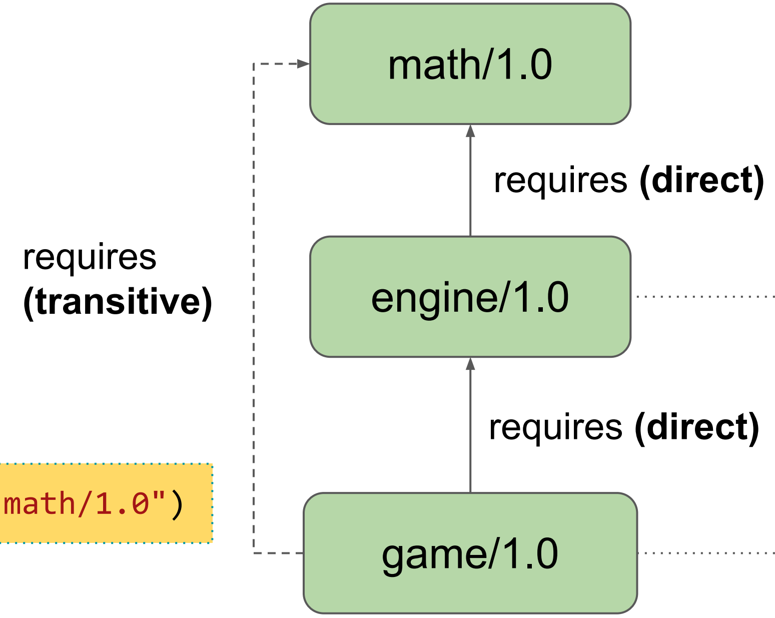
engine/conanfile.py

```
from conan import ConanFile


class Engine(ConanFile):
    name = "engine"
    version = "1.0"

    def requirements(self):
        self.requires("math/1.0",
                      headers=True, libs=False)
```

math-config.cmake

```
~~set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)~~
set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)
```

Conan 2.0 proposal: Direct vs. transitive dependencies

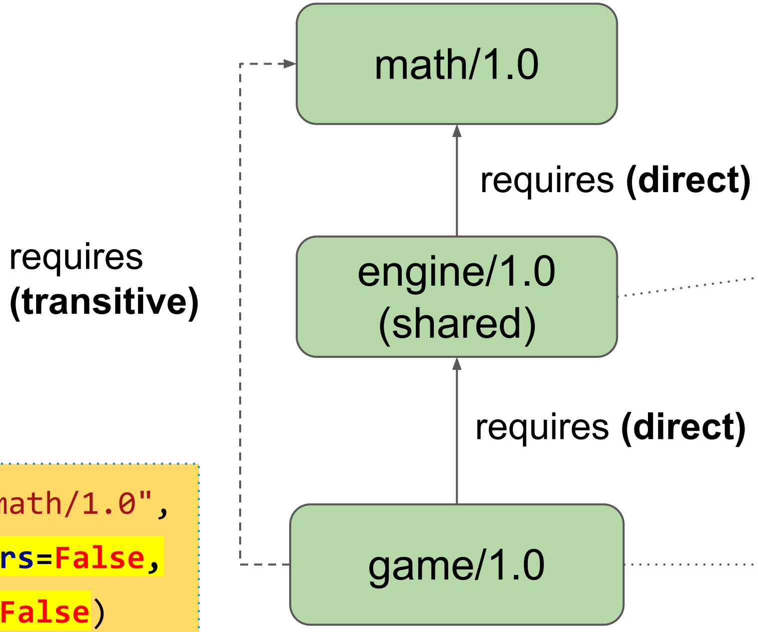


```
self.requires("math/1.0")
```

```
class Engine(ConanFile):
    def requirements(self):
        self.requires("math/1.0")
```

```
class Game(ConanFile):
    def requirements(self):
        self.requires("engine/1.0")
```

Linkage requirements propagation



```
class Engine(ConanFile):
    def requirements(self):
        self.requires("math/1.0",
            headers=True, libs=True,
            transitive_libs=False)
```

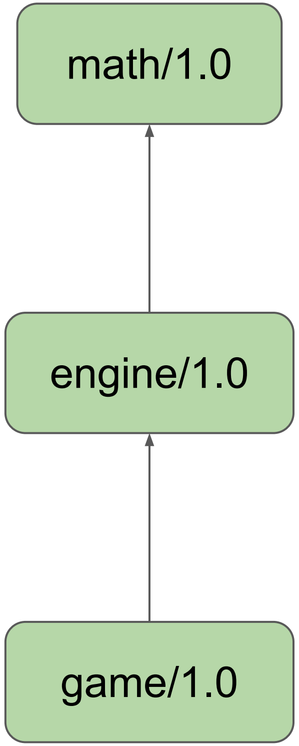
```
class Game(ConanFile):
    def requirements(self):
        self.requires("engine/1.0",
            headers=True, libs=True)
```

```
self.requires("math/1.0",
          headers=False,
          libs=False)
```

math-config.cmake

```
~~set_property(TARGET math::math PROPERTY INTERFACE_LINK_LIBRARIES ...)~~
~~set_property(TARGET math::math PROPERTY INTERFACE_INCLUDE_DIRECTORIES ...)~~
```

Package Types



math/conanfile.py

```
class Math(ConanFile):
    name = "math"
    version = "1.0"
    package_type = "static-library"
    # OR options = {"shared": [True, False]}
```

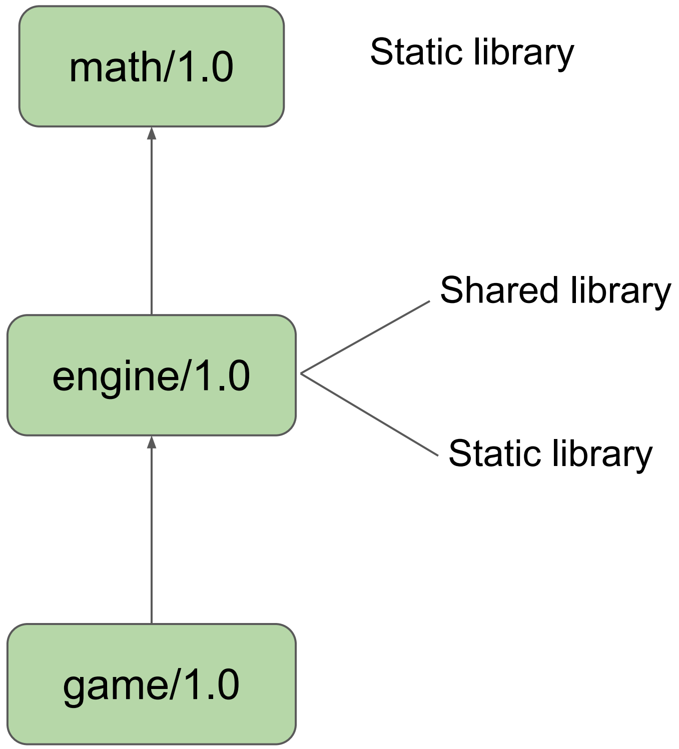
engine/conanfile.py

```
class Engine(ConanFile):
  package_type = "shared-library"
  # OR options = {"shared": [True, False]}
  def requirements(self):
    self.requires("math/1.0")
```

game/conanfile.py

```
class Game(ConanFile):
  package_type = "application"
  def requirements(self):
    self.requires("engine/1.0")
```

Demo


math/1.0
Static library
engine/1.0
Shared library
Static library
game/1.0

Dependency graph 2.0

- Correct linkage requirements
- Correct header visibility
- Possible hidden/private dependencies
- and many more (ACCU 2022)

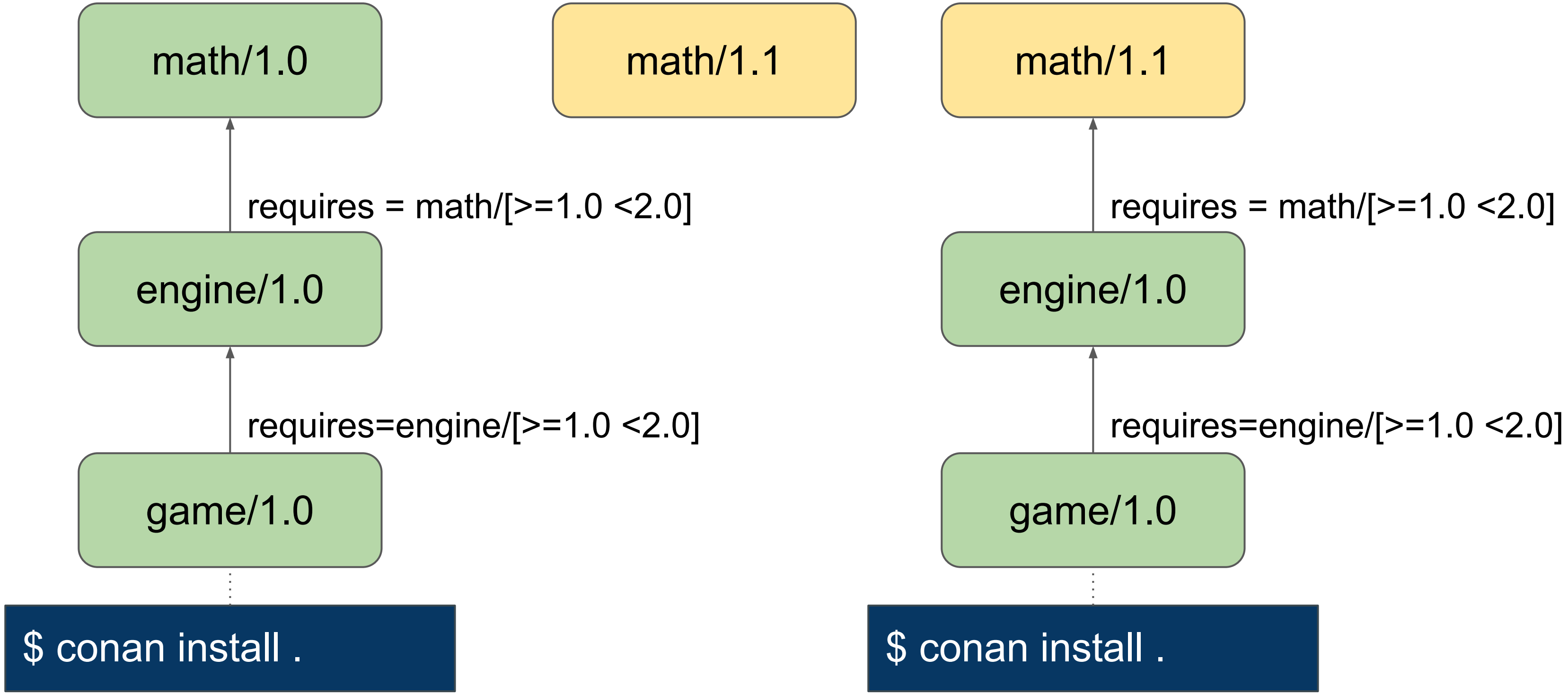
Among different build systems!

Compatible “requires” syntax with 1.X

JFrog | CONAN C/C++ Package Manager

2. Repeating yourself

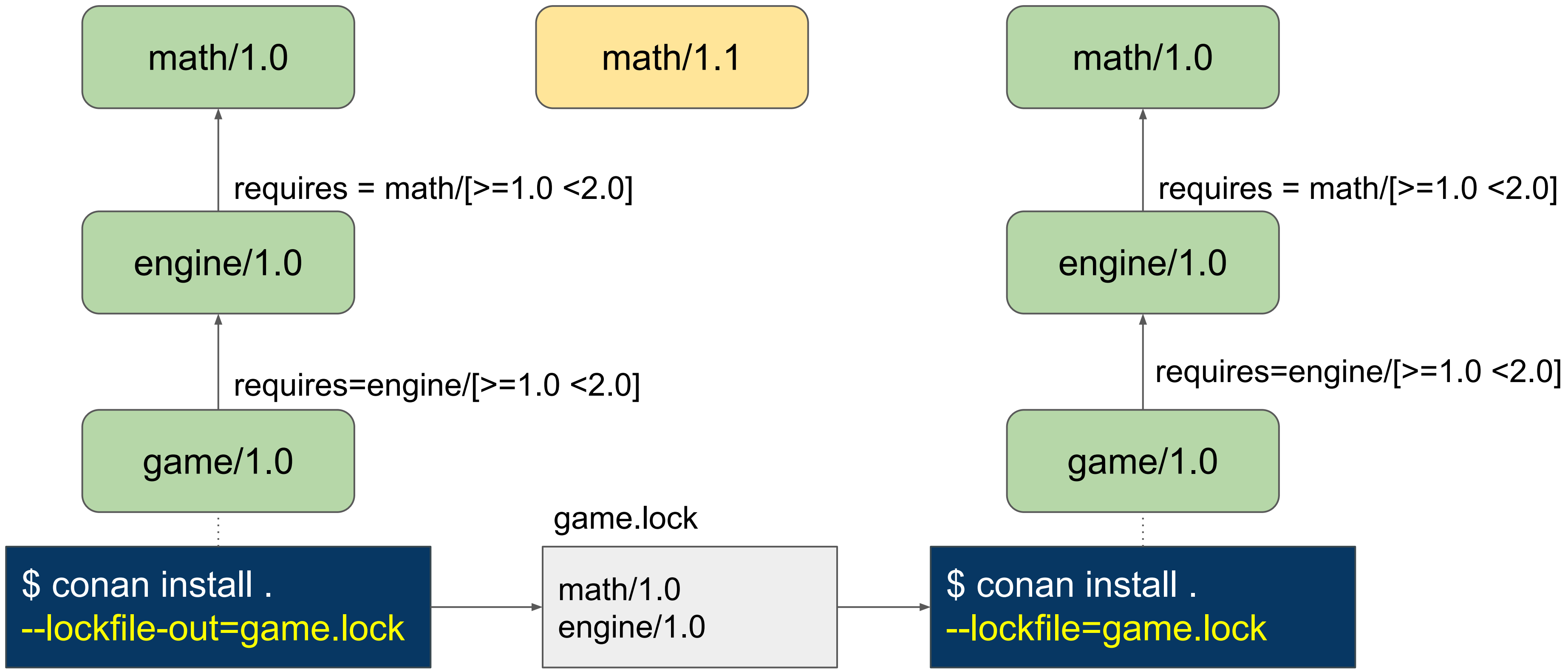
Reproducible dependencies: the problem


math/1.0
math/1.1
math/1.1
requires = math/[>=1.0 <2.0]
requires = math/[>=1.0 <2.0]
engine/1.0
engine/1.0
requires=engine/[>=1.0 <2.0]
requires=engine/[>=1.0 <2.0]
game/1.0
game/1.0

```
$ conan install .
```

```
$ conan install .
```

Reproducible dependencies: Lockfiles


math/1.0
math/1.1
math/1.0
requires = math/[>=1.0 <2.0]
requires = math/[>=1.0 <2.0]
engine/1.0
engine/1.0
requires=engine/[>=1.0 <2.0]
requires=engine/[>=1.0 <2.0]
game/1.0
game/1.0
game.lock
math/1.0
engine/1.0

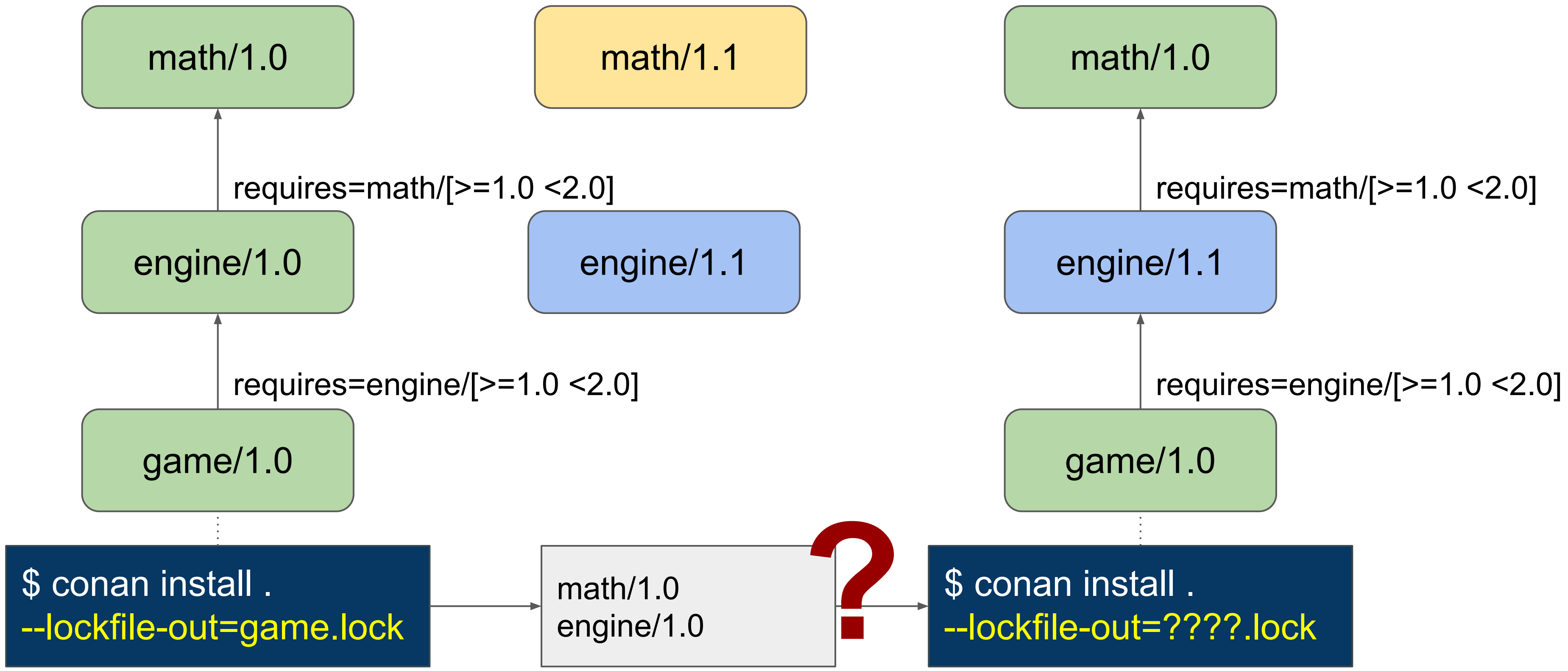
```
$ conan install . --lockfile-out=game.lock
```

```
$ conan install . --lockfile=game.lock
```

Used feature

- 10% of issues last 2.5 years are lockfile related
- Decision tree:
  - Bump “requires = dep/xxx” versions of consumers
  - Use version ranges or revisions
    - Move forward aggressively
    - Lockfiles
- Estimate:
  - 25-40% use lockfiles
  - Demand > 75%

Unleashing the lockfiles for CI


math/1.0
math/1.1
math/1.0
requires=math/[>=1.0 <2.0]
requires=math/[>=1.0 <2.0]
engine/1.0
engine/1.1
engine/1.1
requires=engine/[>=1.0 <2.0]
requires=engine/[>=1.0 <2.0]
game/1.0
game/1.0
math/1.0
engine/1.0
?

```
$ conan install .
--lockfile-out=game.lock
```

```
$ conan install .
--lockfile-out=????.lock
```

Welcome Enterprise DevOps for C and C++

- Enterprise escale can be high
- Enterprise/domain requirements can be challenging
- Continuous Integration at scale is critical
- Thinking beyond package and dependency management
    - Programming over time => SW engineering (T. Winters)
    - Dependency and Package management over time => DevOps

Lockfiles 2.0

1.X

```
"0": {
      "ref": "engine/1.0#fd66..93b7",
      "requires": ["1"],
       },
"1": {"ref": "math/1.0#02fc..3729",
     }
```

2.0

```
"requires": [
  "math/1.0#02fc..3729",
  "engine/1.0#fd66..93b7"
],
"build_requires": [],
"python_requires": []
```

Demo

Lockfiles 2.0

- One lockfile for all configurations
- Easily mutable
- Easily understandable
- Fully strict and partial modes
- Easily mergeable
- Manual commands to modify (override)
- Possible to use multi-project
- Code in codebase 10x shorter
- **Game changer for CI at scale**

3. Building a dam

Extremely opinionated ecosystem

They: I want track compatibility with different systems

Us: Great, what’s important to you to do that?

Person 1: We make the distinction based on C runtime implementation

Person 2: Not really, we just care about the different linux distributions

Person 3: But it is easy, why don’t you just put the exact libc version in the settings?!

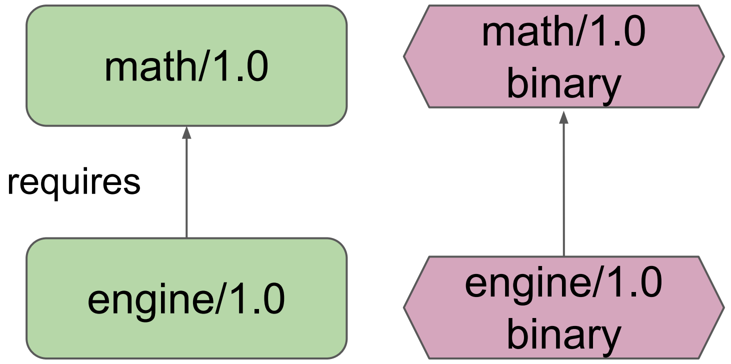
Us: It is easy that they’d never be compatible, different versions means different package_ids, more rebuilding every single time.

…

Person 4: What about windows runtimes?

Binary Compatibility

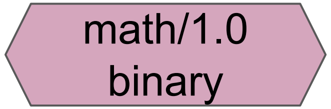
What exactly does this mean? We'll depends who you ask to let me explain the perspective of Conan and how it images packages



Compatible if I change compiler version???

Binary Compatibility

Binary packages each have unique ID regardless of compatibility

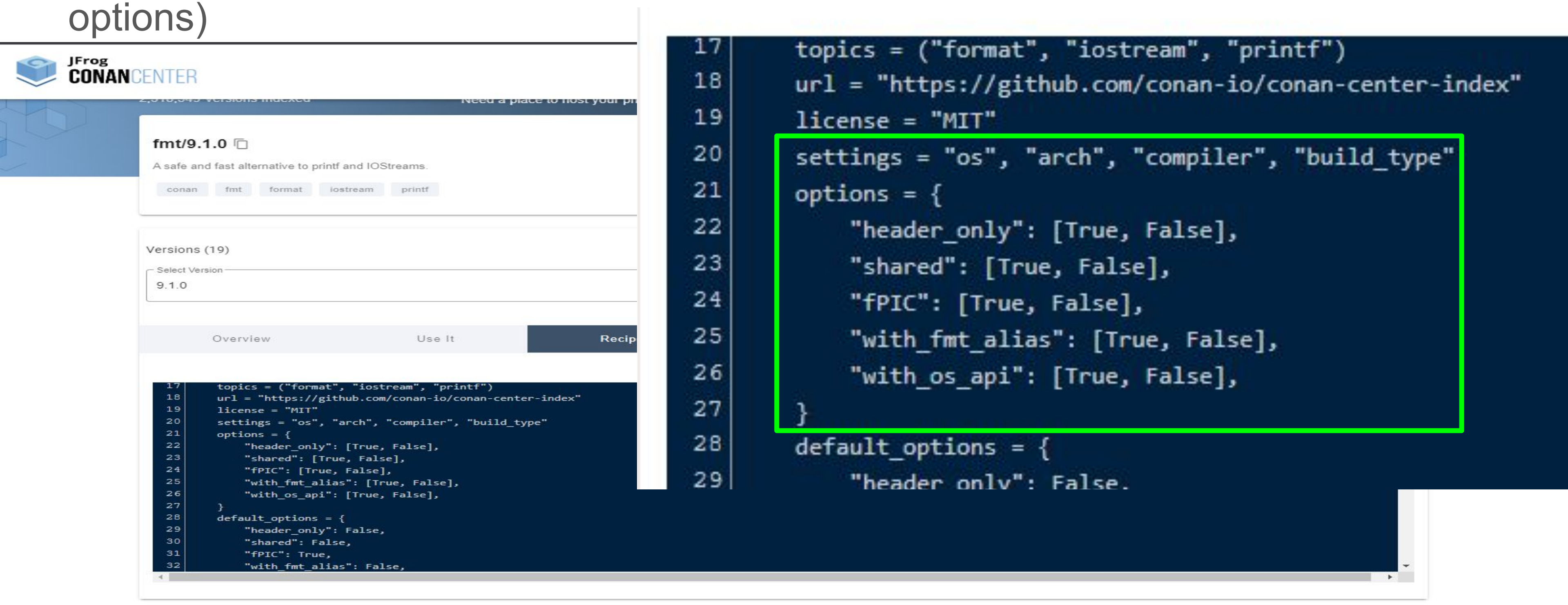


Package_ID: 6af9cc7cb931c5ad942174fd7838eb655717c709

Different configurations – match exactly the same settings (must be compatible) – except when it’s not…

Binary Compatibility

Packages IDs are computer from the binary model of the recipe (settings and options)



fmt/9.1.0

A safe and fast alternative to printf and IOStreams.

conan fmt format iostream printf

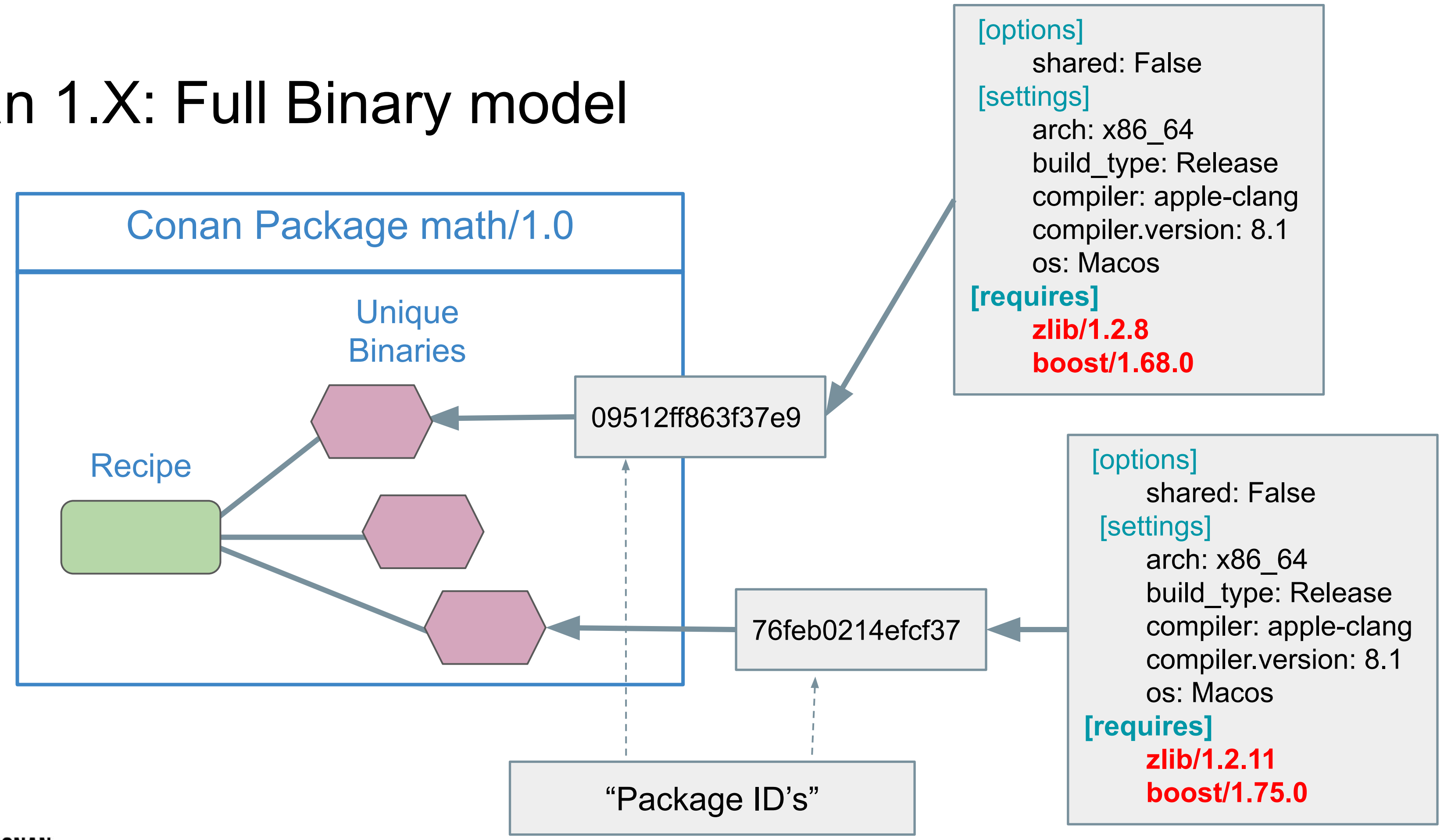
Versions (19)

Select Version 9.1.0

```
    topics = ("format", "iostream", "printf")
    url = "https://github.com/conan-io/conan-center-index"
    license = "MIT"
    settings = "os", "arch", "compiler", "build_type"
    options = {
        "header_only": [True, False],
        "shared": [True, False],
        "fPIC": [True, False],
        "with_fmt_alias": [True, False],
        "with_os_api": [True, False],
    }
    default_options = {
        "header_only": False,
        "shared": False,
        "fPIC": True,
        "with_fmt_alias": False,
```

```
    topics = ("format", "iostream", "printf")
    url = "https://github.com/conan-io/conan-center-index"
    license = "MIT"
    settings = "os", "arch", "compiler", "build_type"
    options = {
        "header_only": [True, False],
        "shared": [True, False],
        "fPIC": [True, False],
        "with_fmt_alias": [True, False],
        "with_os_api": [True, False],
    }
    default_options = {
        "header_only": False,
```

Conan 1.X: Full Binary model

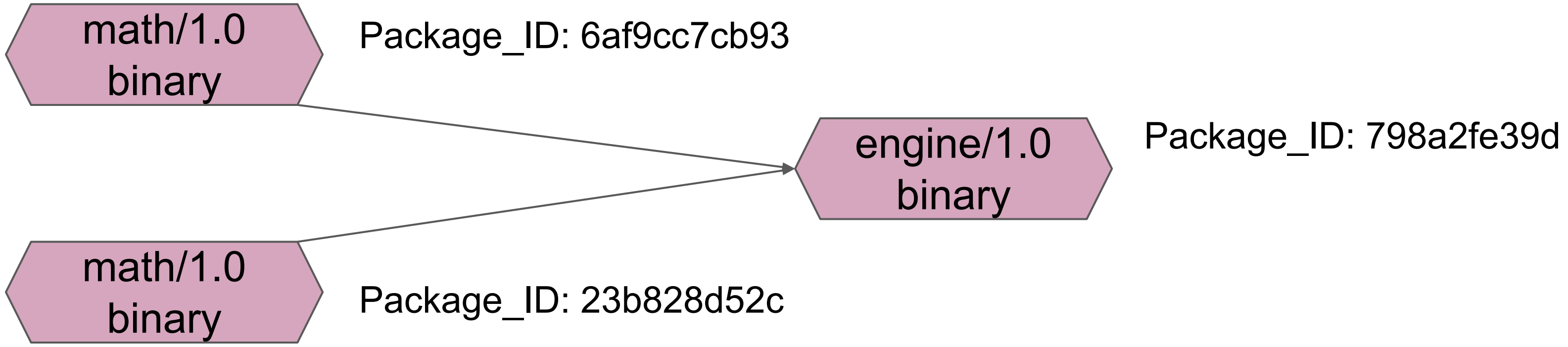


```
[options]
    shared: False
[settings]
    arch: x86_64
    build_type: Release
    compiler: apple-clang
    compiler.version: 8.1
    os: Macos
[requires]
    zlib/1.2.8
    boost/1.68.0
```

```
[options]
    shared: False
[settings]
    arch: x86_64
    build_type: Release
    compiler: apple-clang
    compiler.version: 8.1
    os: Macos
[requires]
    zlib/1.2.11
    boost/1.75.0
```

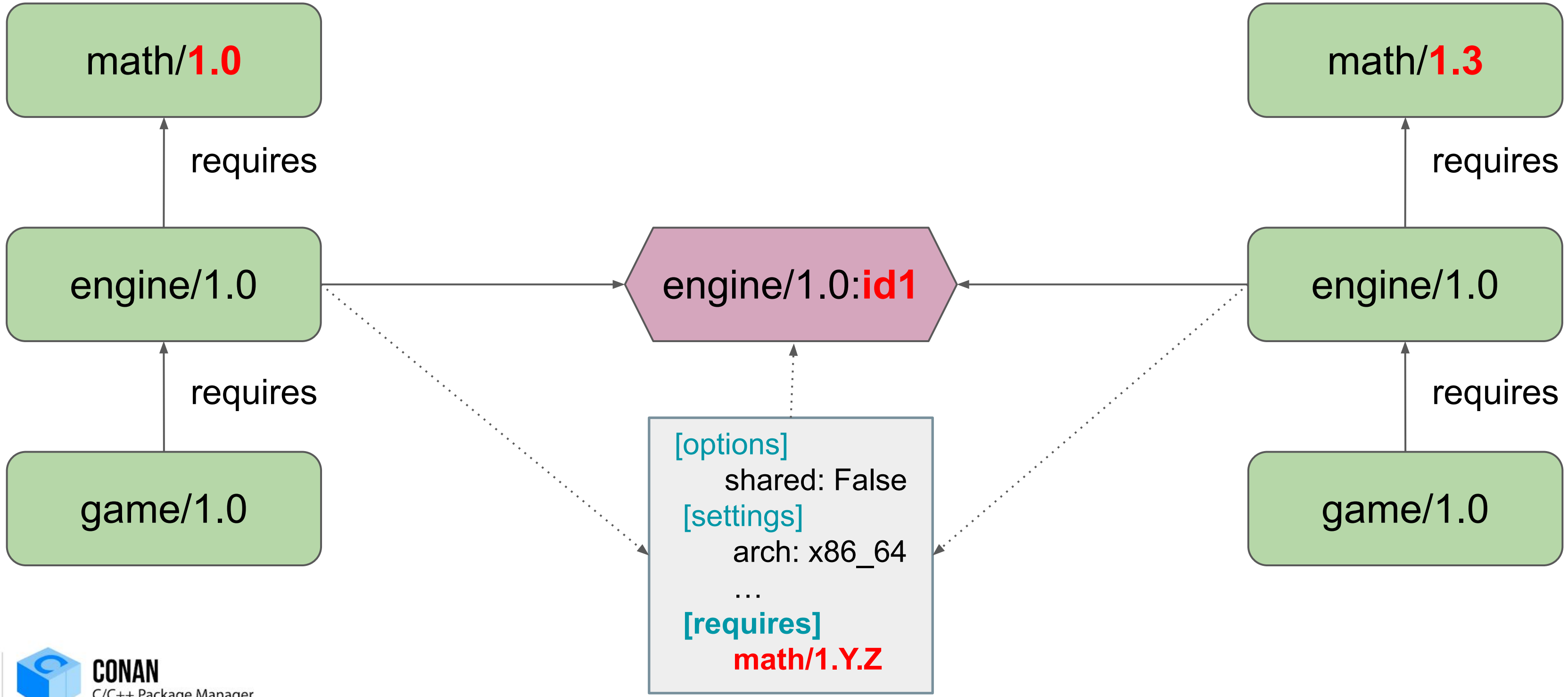
Binary Compatibility

So compatibility in Conan means different package IDs and be interchangeable and still result in a valid final binary



Different inputs – same output

Conan 1.X default package_id_mode = semver



```
[options]
    shared: False
[settings]
    arch: x86_64
    …
[requires]
    math/1.Y.Z
```

Shared library linking static

math2.cpp

```cpp
int add(int a, int b){
  return a + b;
}
```

math2.lib

```
?add@@YAHHH@Z (int __cdecl add(int,int)):
  ...
  ...00011: 03 C8             add        ecx,eax
  ...
```

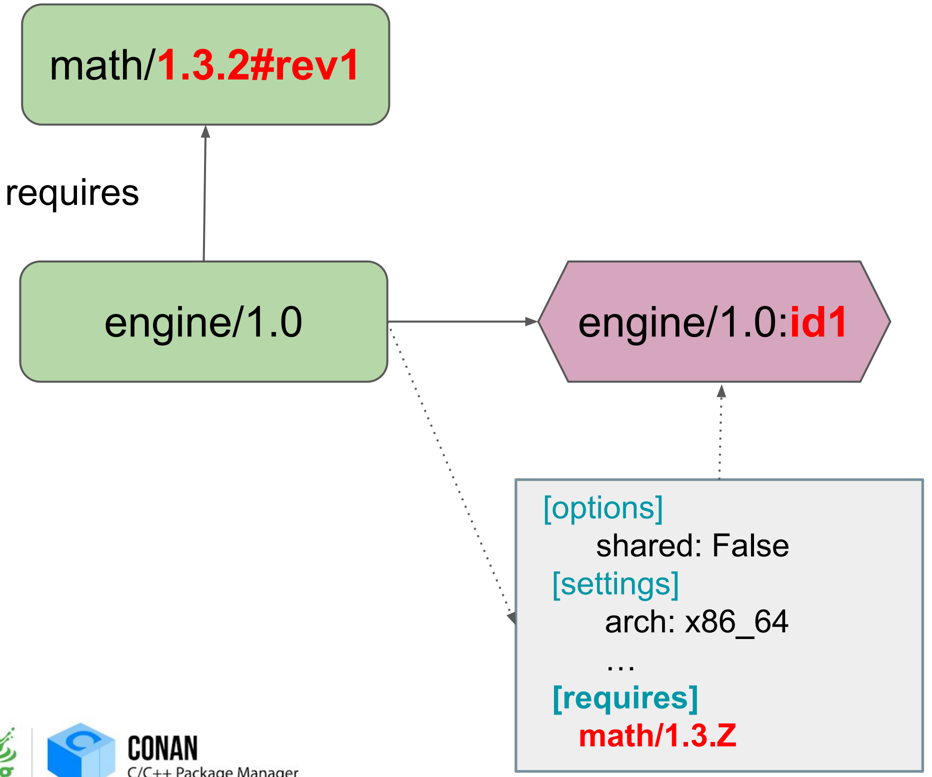
engine.cpp

```cpp
#include "math2.h"
int move3d(int x, int y, int z){
    return add(x, add(y, z));
}
```

engine.dll

```
?move3d@@YAHHHH@Z (int __cdecl move3d(int,int,int)):
  ...
  ...00021: 8B 4C 24 30       mov        ecx,dword ptr [rsp+30h]
  ...00025: 8B 54 24 40       mov        edx,dword ptr [rsp+40h]
  ...00029: 8B 4C 24 38       mov        ecx,dword ptr [rsp+38h]
  ...0002D: E8 00 00 00 00    call       ?add@@YAHHH@Z
  ...00032: 8B D0             mov        edx,eax
  ...00034: 8B 4C 24 30       mov        ecx,dword ptr [rsp+30h]
  ...00038: E8 00 00 00 00    call       ?add@@YAHHH@Z
?add@@YAHHH@Z (int __cdecl add(int,int)):
  ...
  ...00011: 03 C8             add        ecx,eax
```

Conan 2.X default package_id modes: Non embed

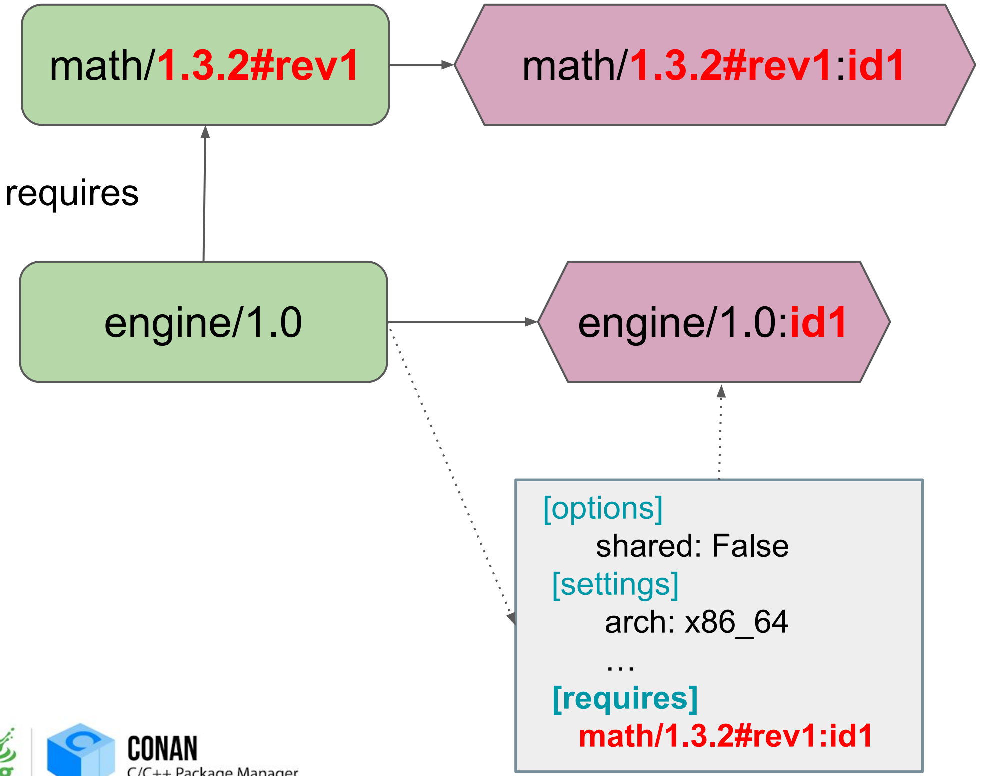


```
[options]
    shared: False
[settings]
    arch: x86_64
    …
[requires]
  math/1.3.Z
```

non_embed_mode (=minor)

- app→ shared
- shared → shared
- static → static

Conan 2.X default package_id modes: Embed



```
[options]
    shared: False
[settings]
    arch: x86_64
    …
[requires]
  math/1.3.2#rev1:id1
```

embed_mode (=full)

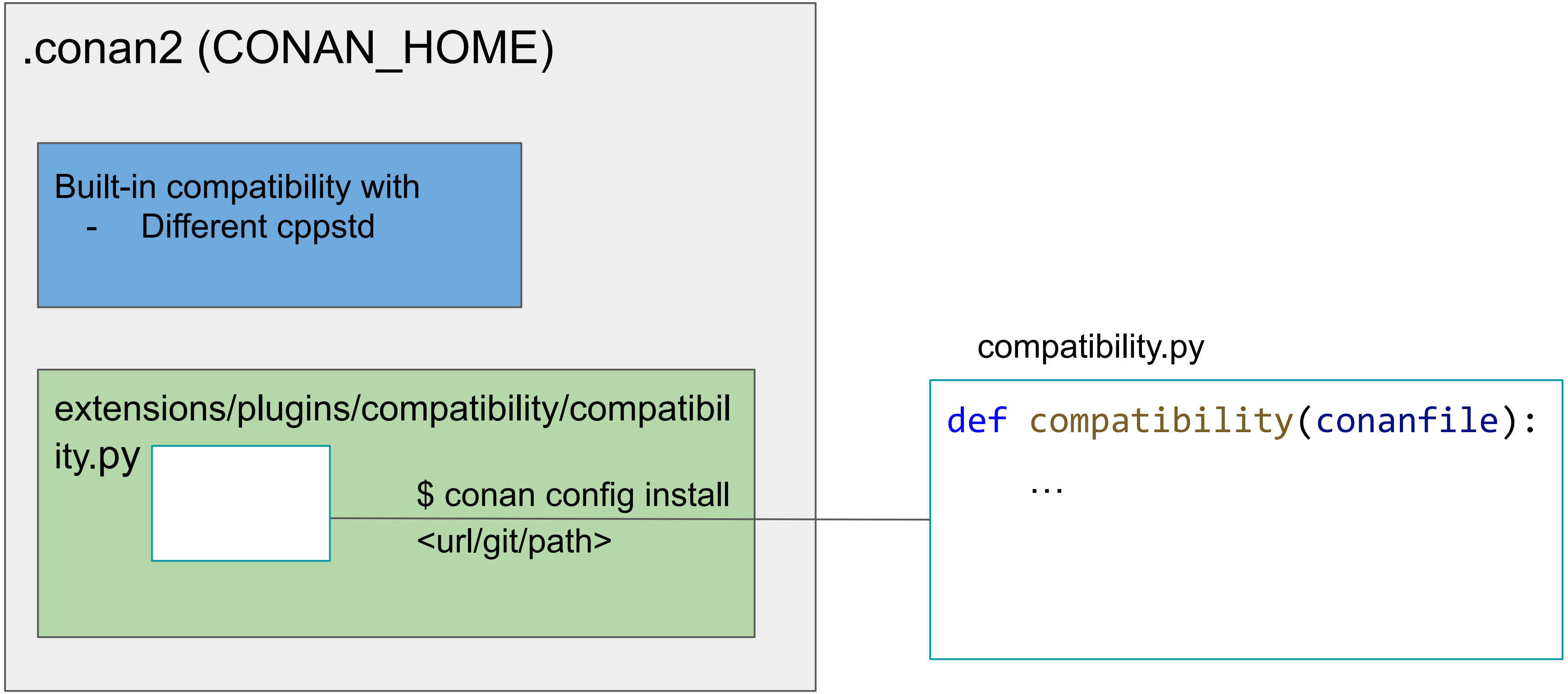
- app→ static
- app → header
- shared → static
- shared → header
- static → header

Configuring package_id

global.conf

```
core.package_id:default_unknown_mode = semver_mode
core.package_id:default_non_embed_mode = minor_mode
core.package_id:default_embed_mode = full_mode
core.package_id:default_python_mode = minor_mode
core.package_id:default_build_mode = None
```

Compatibility Plugin


.conan2 (CONAN_HOME)
Built-in compatibility with
- Different cppstd
extensions/plugins/compatibility/compatibility.py
$ conan config install <url/git/path>
compatibility.py

```python
def compatibility(conanfile):
    …
```

Demo

Conclusions

New graph

New plugin extensions

New deployers

New binary compatibility

Multi-revision cache

package_id

Lockfiles

New configuration and environment

Package immutability optimizations

… and many more

JFrog | CONAN C/C++ Package Manager

https://docs.conan.io/en/2.0/whatsnew.html

Conclusion

pip install conan==2.0.0

https://conan.io