Conan 2.0

Dipl.-Math. Holger Detering

2023-07-12

Agenda

Agenda

Agenda

- Paketmanager für C und C++

- Paketmanager für C und C++
- dezentral

- Paketmanager für C und C++
- dezentral
- Open Source (MIT)

- Paketmanager für C und C++
- dezentral
- Open Source (MIT)
- seit Ende 2016 (Release 1.0: Januar 2018)

- Python

- Python
- Open Source[1]

[1] https://github.com/conan-io/conan

- Python
- Open Source[1]
- Installation:

```
pip install conan
```

[1] https://github.com/conan-io/conan

- Python
- Open Source[1]
- Installation:

```
pip install --user conan
```

[1] https://github.com/conan-io/conan

- Python
- Open Source[1]
- Installation:

```
pip install --user --upgrade conan
```

[1] https://github.com/conan-io/conan

Lifecoding: Installation des Clients

- Conancenter[2]

[2] https://conan.io/center/

- Conancenter[2]
- JFrog Artifactory Community Edition for C/C++ [3]

[2] https://conan.io/center/
[3] https://conan.io/downloads.html

- Conancenter[2]
- JFrog Artifactory Community Edition for C/C++ [3]
- Conan Server (zum Testen und für kleine Teams)[4]

[2] https://conan.io/center/
[3] https://conan.io/downloads.html
[4] https://docs.conan.io/2/reference/conan_server.html

Lifecoding: JFrog Artifactory Community Edition

- conanfile.txt

- conanfile.txt
  - INI Format

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen
  - [requires] – Abhängigkeiten

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen
  - [requires] – Abhängigkeiten
  - [tool-requires] – Abhängkeiten (Build-Werkzeuge)

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen
  - [requires] – Abhängigkeiten
  - [tool-requires] – Abhängkeiten (Build-Werkzeuge)
  - [generators] – cmake, virtualenv, ...

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen
  - [requires] – Abhängigkeiten
  - [tool-requires] – Abhängkeiten (Build-Werkzeuge)
  - [generators] – cmake, virtualenv, ...
  - [options] – Optionen für die Abhängigkeiten

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen
  - [requires] – Abhängigkeiten
  - [tool-requires] – Abhängkeiten (Build-Werkzeuge)
  - [generators] – cmake, virtualenv, ...
  - [options] – Optionen für die Abhängigkeiten
- conanfile.py

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen
  - [requires] – Abhängigkeiten
  - [tool-requires] – Abhängkeiten (Build-Werkzeuge)
  - [generators] – cmake, virtualenv, ...
  - [options] – Optionen für die Abhängigkeiten
- conanfile.py
  - Python Skript

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen
  - [requires] – Abhängigkeiten
  - [tool-requires] – Abhängkeiten (Build-Werkzeuge)
  - [generators] – cmake, virtualenv, ...
  - [options] – Optionen für die Abhängigkeiten
- conanfile.py
  - Python Skript
  - Generieren (und Konsumieren) von Conan Paketen

- conanfile.txt
  - INI Format
  - Konsumieren von Conan Paketen
  - [requires] – Abhängigkeiten
  - [tool-requires] – Abhängkeiten (Build-Werkzeuge)
  - [generators] – cmake, virtualenv, ...
  - [options] – Optionen für die Abhängigkeiten
- conanfile.py
  - Python Skript
  - Generieren (und Konsumieren) von Conan Paketen
  - Verwendet Conans Python API

Lifecoding: Beispielprojekt

Agenda

Die Abhängigkeiten in C und C++ sind kompliziert

Das „require“ Modell ist zu einfach:

Die Abhängigkeiten in C und C++ sind kompliziert

Das „require“ Modell ist zu einfach:

- Wie Compiler und Linker Flags weitergereicht werden, hängt vom Typ der Libraries ab (statisch / shared / header-only).

Die Abhängigkeiten in C und C++ sind kompliziert

Das „require“ Modell ist zu einfach:

- Wie Compiler und Linker Flags weitergereicht werden, hängt vom Typ der Libraries ab (statisch / shared / header-only).
- Müssen die Sichtbarkeiten der Header weitergereicht werden?

Die Abhängigkeiten in C und C++ sind kompliziert

Das „require“ Modell ist zu einfach:

- Wie Compiler und Linker Flags weitergereicht werden, hängt vom Typ der Libraries ab (statisch / shared / header-only).
- Müssen die Sichtbarkeiten der Header weitergereicht werden?
- Was passiert, wenn im Abhängigkeitsbaum unterschiedliche Versionen einer statischen gebauten Library verwendet werden, die in Shared Libraries versteckt sind?

C und C++ werden in sehr unterschiedlichen Konstellationen eingesetzt

- Die Einsatzgebiete für C und C++ sind Legion.

C und C++ werden in sehr unterschiedlichen Konstellationen eingesetzt

- Die Einsatzgebiete für C und C++ sind Legion.
  $\Rightarrow$ Vielfalt an Anwendungsfällen, Entwicklungsprozessen, Complianceregeln, Policies...

C und C++ werden in sehr unterschiedlichen Konstellationen eingesetzt

- Die Einsatzgebiete für C und C++ sind Legion.
  $\Rightarrow$ Vielfalt an Anwendungsfällen, Entwicklungsprozessen, Complianceregeln, Policies...
- Conan 1.X nur sehr begrenzt erweiterbar – Anpassung an eigene Bedürfnisse kompliziert

C und C++ werden in sehr unterschiedlichen Konstellationen eingesetzt

- Die Einsatzgebiete für C und C++ sind Legion.
  $\Rightarrow$ Vielfalt an Anwendungsfällen, Entwicklungsprozessen, Complianceregeln, Policies...
- Conan 1.X nur sehr begrenzt erweiterbar – Anpassung an eigene Bedürfnisse kompliziert
- Python API praktisch wenig hilfreich

C++ und DevOps

- Bedürfnis, Abhängigkeitsgraph zu einem bestimmten Zeitpunkt exakt reproduzieren

C++ und DevOps

- Bedürfnis, Abhängigkeitsgraph zu einem bestimmten Zeitpunkt exakt reproduzieren
- CI / CD Unterstützung

- 2.0 im Februar 2023

- 2.0 im Februar 2023
- aktuell: 2.07

Conan Client

- Konsistentere Schnittstelle der Conan Befehle

Conan Client

- Konsistentere Schnittstelle der Conan Befehle
- Verschiedene Ausgabeformate und Terminalumleitung

Conan Client

- Konsistentere Schnittstelle der Conan Befehle
- Verschiedene Ausgabeformate und Terminalumleitung
- Aufgeräumtere Ausgabe

Conan Client

- Konsistentere Schnittstelle der Conan Befehle
- Verschiedene Ausgabeformate und Terminalumleitung
- Aufgeräumtere Ausgabe
- Verbesserte, strukturierte machschinenlesbare Ausgabe (JSON, CI)

Build System Integration

- Transparente Integration mit CMake

Build System Integration

- Transparente Integration mit CMake
- CMake Presets: CMake Aufruf einfacher

Build System Integration

- Transparente Integration mit CMake
- CMake Presets: CMake Aufruf einfacher
- Gleiches Modell auch für Autotools, Xcode und MSBuild

Build System Integration

- Transparente Integration mit CMake
- CMake Presets: CMake Aufruf einfacher
- Gleiches Modell auch für Autotools, Xcode und MSBuild
- generate() Methode

Modell des Abhängigkeitsbaumes[5]

- Requirements Traits: Wie und wann werden die verschiedenen Teile einer Abhängigkeit weitergegeben? (Includepfade, Libraries, zur Build- oder zur Laufzeit? transitiv?)

[5] https://www.youtube.com/watch?v=kKGglzm5ous

Modell des Abhängigkeitsbaumes[5]

- Requirements Traits: Wie und wann werden die verschiedenen Teile einer Abhängigkeit weitergegeben? (Includepfade, Libraries, zur Build- oder zur Laufzeit? transitiv?)
- Paket Typen: static / shared / header-only Libraries, Applikationen

[5] https://www.youtube.com/watch?v=kKGglzm5ous

Modell des Abhängigkeitsbaumes[5]

- Requirements Traits: Wie und wann werden die verschiedenen Teile einer Abhängigkeit weitergegeben? (Includepfade, Libraries, zur Build- oder zur Laufzeit? transitiv?)
- Paket Typen: static / shared / header-only Libraries, Applikationen
- Verbesserte Modellierung der nur für den Build notwendigen Abhängigkeiten

[5] https://www.youtube.com/watch?v=kKGglzm5ous

Modell des Abhängigkeitsbaumes[5]

- Requirements Traits: Wie und wann werden die verschiedenen Teile einer Abhängigkeit weitergegeben? (Includepfade, Libraries, zur Build- oder zur Laufzeit? transitiv?)
- Paket Typen: static / shared / header-only Libraries, Applikationen
- Verbesserte Modellierung der nur für den Build notwendigen Abhängigkeiten
- Verbesserte Unterstützung beim Cross-Compiling

[5] https://www.youtube.com/watch?v=kKGglzm5ous

Erweiterbarkeit

- Lokale Settings Datei (settings_user.yml); keine Anpassung an globaler settings.yml nötig

Erweiterbarkeit

- Lokale Settings Datei (settings_user.yml); keine Anpassung an globaler settings.yml nötig
- Kommando Wrapper Erweiterung, die self.run() Aufrufe abfängt (z.B. für verteiltes Bauen)

Erweiterbarkeit

- Lokale Settings Datei (settings_user.yml); keine Anpassung an globaler settings.yml nötig
- Kommando Wrapper Erweiterung, die self.run() Aufrufe abfängt (z.B. für verteiltes Bauen)
- Benutzerdefinierte Conan Befehle (*custom commands*)

Erweiterbarkeit

- Lokale Settings Datei (settings_user.yml); keine Anpassung an globaler settings.yml nötig
- Kommando Wrapper Erweiterung, die self.run() Aufrufe abfängt (z.B. für verteiltes Bauen)
- Benutzerdefinierte Conan Befehle (*custom commands*)
- Profile und Konfigurationen unterstützen Jinja2 Templates

Erweiterbarkeit

- Lokale Settings Datei (settings_user.yml); keine Anpassung an globaler settings.yml nötig
- Kommando Wrapper Erweiterung, die self.run() Aufrufe abfängt (z.B. für verteiltes Bauen)
- Benutzerdefinierte Conan Befehle (*custom commands*)
- Profile und Konfigurationen unterstützen Jinja2 Templates
- Python API der Conan Befehle öffentlich

Dokumentation

- Dokumentation komplett neu geschrieben

Dokumentation

- Dokumentation komplett neu geschrieben
- Tutorial

Dokumentation

- Dokumentation komplett neu geschrieben
- Tutorial
- Beispiele aufgeräumt

Dokumentation

- Dokumentation komplett neu geschrieben
- Tutorial
- Beispiele aufgeräumt
- Migrationsleitfaden Conan 1.X → 2.0

Was noch?

- CONAN_HOME ist jetzt ~/.conan2 (Conan 1.X: ~/.conan)

Was noch?

- CONAN_HOME ist jetzt ~/.conan2 (Conan 1.X: ~/.conan)
- Revisionen defaultmäßig aktiv

Was noch?

- CONAN_HOME ist jetzt ~/.conan2 (Conan 1.X: ~/.conan)
- Revisionen defaultmäßig aktiv
- lokaler Cache: Mehrere Revisionen eines Paketes, verkürzte Pfade

Was noch?

- CONAN_HOME ist jetzt ~/.conan2 (Conan 1.X: ~/.conan)
- Revisionen defaultmäßig aktiv
- lokaler Cache: Mehrere Revisionen eines Paketes, verkürzte Pfade
- Lockfiles

Was noch?

- CONAN_HOME ist jetzt ~/.conan2 (Conan 1.X: ~/.conan)
- Revisionen defaultmäßig aktiv
- lokaler Cache: Mehrere Revisionen eines Paketes, verkürzte Pfade
- Lockfiles
- Neue Konfigurationsdateien

Was noch?

- CONAN_HOME ist jetzt ~/.conan2 (Conan 1.X: ~/.conan)
- Revisionen defaultmäßig aktiv
- lokaler Cache: Mehrere Revisionen eines Paketes, verkürzte Pfade
- Lockfiles
- Neue Konfigurationsdateien
- Paket Signing

Was noch?

- CONAN_HOME ist jetzt ~/.conan2 (Conan 1.X: ~/.conan)
- Revisionen defaultmäßig aktiv
- lokaler Cache: Mehrere Revisionen eines Paketes, verkürzte Pfade
- Lockfiles
- Neue Konfigurationsdateien
- Paket Signing
- ...

ConanCenter[6]

- >1500 Conan Rezepte

[6] https://github.com/conan-io/conan-center-index

ConanCenter[6]

- >1500 Conan Rezepte
- die meisten (wichtigsten) funktionieren bereits mit Conan 2

[6] https://github.com/conan-io/conan-center-index

Lifecoding: Beispielprojekt mit Conan 2

Fragen?

- Homepage: `https://conan.io`
- Download: `https://conan.io/downloads.html`
- Doku: `https://docs.conan.io/`
- Slack: #conan auf `https://cpplang.slack.com`
- Include<C++> discord: conan auf `https://www.includecpp.org/discord`
- Github (Client): `https://github.com/conan-io/conan`
- ConanCenter: `https://conan.io/center`
- Github (ConanCenter): `https://github.com/conan-io/conan-center-index`
- Cheat-Sheet Vergleich: `https://blog.conan.io/2023/06/07/New-Cheat-Sheet-For-Conan-2.html`
- Diego Rodriguez-Losada: Advanced Dependencies Model in Conan 2.0 (ACCU 2022): `https://www.youtube.com/watch?v=kKGglzm5ous`