Conan C/C++ package manager

START HERE

`conan profile detect`
Create default profile

`conan new cmake_lib -d name=<n> -d version=<v>`
Scaffold a package

`conan install . --build=missing`
Fetch or build deps

`conan build .`
Build your project

`conan create .`
Package into cache

`conan search "<pkg>*" -r conancenter`
Find a package

`conan list "<pkg>/*"`
List cached versions

`conan remote list`
Show remotes

Recipe conanfile.py → `conan create` → **Local cache** ~/.conan2 → `conan upload` → **Remote** conancenter, Artifactory

CONCEPTS

Conan 2 reads a conanfile.txt or conanfile.py; a profile picks compiler and settings, generators feed your build system.

Settings, options and conf given on the command line override the profile; use -pr:b for the build machine and -pr:h for the target.

Binaries are keyed by a package ID hashed from settings and options; any difference means a different binary.

PROJECT SETUP

`conan profile detect`
Generate the default profile
Run once per machine; edit the file to change compiler or build_type.

`conan new cmake_lib -d name=<n> -d version=<v>`
Scaffold a CMake library recipe
Other templates: cmake_exe, meson_lib, header_lib, basic.

`conan new -h`
List the built-in templates

`conan version`
Print the Conan version

`conan config home`
Print the Conan home folder

`conan config install <url-or-folder>`
Install shared remotes and profiles

INSTALL & BUILD

`conan install .`
Install dependencies of this project
Run in the folder with conanfile.txt or conanfile.py.

`conan install . --build=missing`
Build binaries that are not available

`conan build .`
Install deps and run the recipe's build()

`conan install . -of <dir>`
Write generated files to another folder

`conan install . -g CMakeDeps`
Add a generator

`conan install --requires=<pkg>/<ver>`
Install a package without a recipe

INSTALL & BUILD (cont.)

`conan install . -s build_type=Debug`
Override a setting

`conan install . -o <pkg>/*:shared=True`
Set a package option

`conan install . -c <key>=<value>`
Set a conf value

`conan install . -pr:b <profile> -pr:h <profile>`
Cross-build with two profiles

`conan install . -r <remote>`
Use only one remote

`conan install . -u`
Check remotes for newer recipes

`conan install . -d full_deploy`
Copy dependencies into a folder

CREATE PACKAGES

`conan create .`
Build the recipe and test it
Runs test_package if the folder exists.

`conan create . --build=missing`
Create, building missing deps

`conan create . -tf=""`
Create without running test_package

`conan export .`
Put only the recipe in the cache

`conan export-pkg .`
Package already-built binaries

`conan test test_package <pkg>/<ver>`
Test an existing package

`conan editable add <path>`
Use a local folder instead of the cache

`conan editable list`
Show editable packages

PROFILES

`conan profile list`
List profiles in the cache

`conan profile show`
Show the resolved default profile

`conan profile show -pr <profile> -s build_type=Debug`
Preview a profile plus overrides

`conan profile path <profile>`
Print a profile's file path

PROFILES (cont.)

`conan profile detect --name <profile>`
Detect into a named profile

REMOTES

`conan remote list`
List remotes

`conan remote add <name> <url>`
Add a remote

`conan remote login <remote> <user>`
Log in to a remote

`conan remote auth <pattern>`
Authenticate (CI friendly)
Reads CONAN_LOGIN_USERNAME and CONAN_PASSWORD.

`conan remote remove <remote>`
Delete a remote

`conan remote disable <pattern>`
Turn off matching remotes

SEARCH & LIST

`conan search "<pkg>*"`
Search recipes in all remotes

`conan search "<pkg>*" -r <remote>`
Search one remote

`conan list "<pkg>/*"`
List cached versions

`conan list "<pkg>/<ver>:*"`
List cached binaries

`conan list "<pkg>/*" -r <remote>`
List versions on a remote

`conan list "*" -f json`
Machine-readable listing

`conan list "<pkg>/<ver>:*" -p "os=Linux AND build_type=Release"`
Filter binaries by query

`conan graph info .`
Show the dependency graph

`conan graph info . -f html > graph.html`
Graph as an HTML page

`conan graph explain .`
Why is no binary matching?

`conan graph outdated .`
List newer dependency versions

UPLOAD & CACHE

`conan upload "<pkg>/<ver>" -r <remote>`
Upload recipe and binaries

UPLOAD & CACHE (cont.)

`conan upload "*" -r <remote> -c`
Upload everything, no prompt

`conan upload "<pkg>/<ver>" -r <remote> --only-recipe`
Upload only the recipe

`conan download "<pkg>/<ver>" -r <remote>`
Download without installing

`conan remove "<pkg>/<ver>"`
Delete from the local cache

`conan remove "<pkg>/*" -c`
Remove all versions, no prompt

`conan remove "<pkg>/<ver>:*" -c`
Remove binaries, keep recipe

`conan cache path <pkg>/<ver>`
Print a recipe's cache folder

`conan cache clean`
Drop source and build folders

LOCKFILES

`conan lock create .`
Freeze dependency versions
Later commands pick up conan.lock automatically.

`conan install . -l <file>.lock`
Use a specific lockfile

RECIPES

Dependencies via conanfile.txt

```
[requires]
zlib/1.3.1

[generators]
CMakeDeps
CMakeToolchain

[layout]
cmake_layout
```

CMake project with Conan

```
conan install . --build=missing
cmake --preset conan-release
cmake --build --preset conan-release
```

Package and publish a library

```
conan new cmake_lib -d
name=hello -d version=1.0
conan create .
conan remote add <name> <url>
conan remote login <name>
<user>
conan upload "hello/1.0" -r
<name> -c
```

Debug build of a dependency

```
conan profile detect
conan install . -s
build_type=Debug --
build=missing
conan graph info . -s
build_type=Debug
```