Conan Strikes Back: Easy Migration to Conan 2.0

Evgenii Seliverstov
@theirix

What is Conan?

Quiz? A package and dependency manager guessing game!

- Java – Maven
- Rust – Cargo
- .NET – NuGet
- JavaScript – npm
- Ruby – Gem & Bundler
- Python – Poetry / uv / pipx / pdm
- Perl – CPAN
- Haskell – Cabal
- Elixir – Hex
- C++ – 🤔?

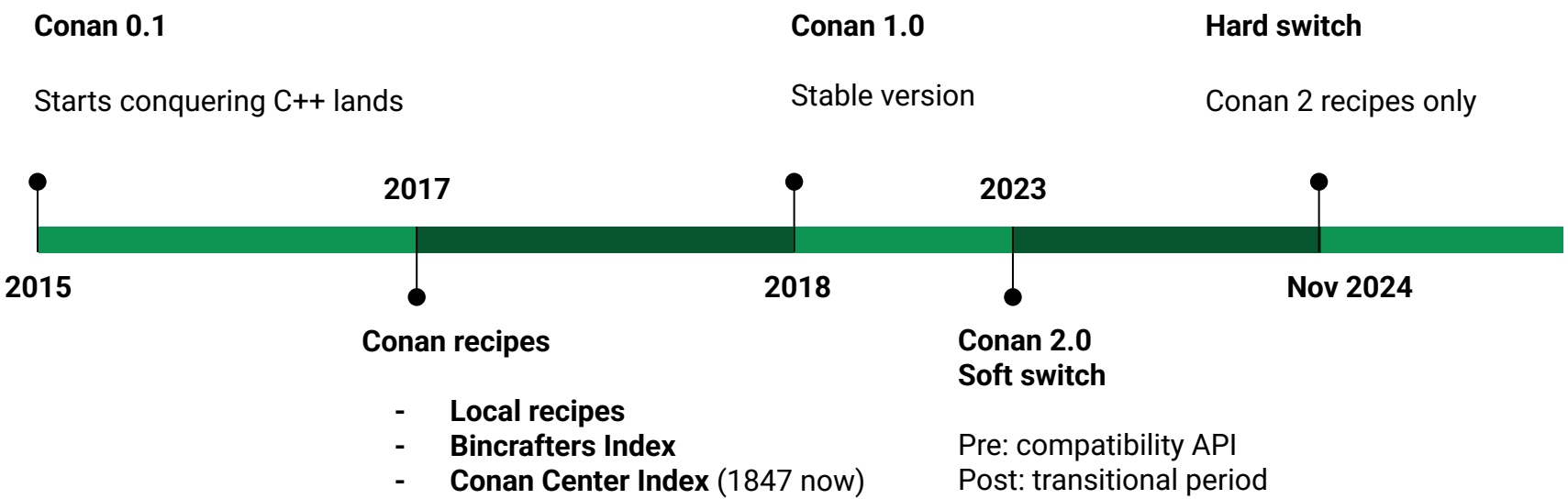
What is Conan?

A cross-platform C++ package and dependency manager.

The state of package managers:

- Until 2016 – dark ages
- 2016 – vcpkg
- 2017 – Conan 1.0

History of the Conan Journey



Consuming packages

Consume packages in an application or a library

`conanfile.txt`

```
[requires]
zstd/1.5.2
openssl/1.1.1q
boost/1.75.0

[generators]
cmake
cmake_find_package

[options]
boost:shared=True
```

`CMakeLists.txt`

```
project(example CXX)

include(${CMAKE_BINARY_DIR}/conanbuildinfo.cmake)
conan_basic_setup(TARGETS)

add_executable(example main.cpp)
target_link_libraries(example
  CONAN_PKG::zstd CONAN_PKG::boost)
```

```
conan install .
cmake -B build -S .
cmake --build build
```

Creating packages

```
class ZstdConan(conans.ConanFile):                          # Simplified Conan 1.0 recipe
    name = "zstd"
    generators = "cmake"
    settings = "os", "arch", "compiler", "build_type"
    options = {"shared": [True, False], "fPIC": [True, False]}

    def source(self):
        pass

    def build(self):
        cmake = conans.CMake(self)
        cmake.definitions["ZSTD_BUILD_SHARED"] = self.options.shared
        cmake.configure()
        cmake.build()

    def package(self):
        conans.CMake(self).install()

    def package_info(self):
        self.cpp_info.libs = conans.tools.collect_libs(self)
        if self.settings.os == "Linux": self.cpp_info.libs.append("pthread")
```

Why Conan 2?

- Better understanding of practices after providing ~2K recipes
  - Hashing package IDs
  - Mostly compatibility blockers
  - CLI rework
  - Virtualenv (aka `activate`)
- Invisible for consumers (no `conan_basic_setup` or `CONAN_PKG`s)
  - CMake toolchains
  - CMake presets
- Better Conan API
  - Python imports
  - Dependency graph
  - Integrations with CI and external systems

Waiting for Conan 2

- New experimental API from Conan libraries for recipes
- Special backported API to 1
- Packages started to transition to support both 1 and 2
- Conan 2.0 released
- Supporting packages for 1 and 2
- Dropping old functionality
- New packages are Conan 2 only

How it works

- Recipes from Conan Center Index support both Conan 1.0 and 2.0 API
- For users:
  - Conan 2 tool works only with Conan 2 packages (same for 1)
  - Conan 1 recipes are still there (different remote)

How it works

- Recipes from Conan Center Index support both Conan 1.0 and 2.0 API
- For users:
  - Conan 2 tool works only with Conan 2 packages (same for 1)
  - Conan 1 recipes are still there (different remote)
- For recipe developers:
  - Conan 1 recipes can be consumed by Conan 1 only
  - Aware recipes can be published to 1 and 2 simultaneously
- Local usage - ~/.conan2 vs ~/.conan

The Conan Way – Consumers

- For users – very simple!
- Change `conanfile.txt` to switch to `CMakeToolchain` generator
- Strip `CMakeLists.txt` from Conan mentions
- Dependencies dropped @ suffix: `zstd/1.5.2` to `zstd/1.5.2`
- Learn new CLI (it's better!)

The Conan Way – Recipes

- For recipes authors – some work
- One example for zstd – check QR at the right
- Use compatibility API (if you have to support Conan 1)
- Change generator from ‘cmake’ to `CMakeToolchain` and `CMakeDeps`
- Specify layout (e.g. `cmake_layout`) to drop all dir handling
- Automatic source patch apply
- Move configuration options – move from `build` to `generate`
- `def package` – significant `copy` rework with new layout
- `def imports` – move to `generate`
- `def package_info` - migrate from `.names` to consolidated `cmake_target_name`

The Conan Way – a zstd diff (1/5)

```
--- a/recipes/zstd/all/conanfile.py
+++ b/recipes/zstd/all/conanfile.py
@@ -1,44 +1,37 @@
-from conans import ConanFile, CMake, tools
+from conan import ConanFile
+from conan.tools.cmake import CMake, CMakeToolchain, cmake_layout
+from conan.tools.files import apply_conandata_patches, collect_libs, copy, export_conandata_patches, get, replace_in_file, rmdir, rm
+from conan.tools.scm import Version
+import glob
 import os
```

The Conan Way – a zstd diff (2/5)

```
    def export_sources(self):
-       self.copy("CMakeLists.txt")
-       for patch in self.conan_data.get("patches", {}).get(self.version, []):
-           self.copy(patch["patch_file"])
+       export_conandata_patches(self)
+
-   @property
-   def _source_subfolder(self):
-       return "source_subfolder"
-   @property
-   def _build_subfolder(self):
-       return "build_subfolder"
+
+   def layout(self):
+       cmake_layout(self, src_folder="src")
```

The Conan Way – a zstd diff (3/5)

```
-   def build(self):
-       self._cmake = CMake(self)
-       self._cmake.definitions["ZSTD_BUILD_PROGRAMS"] = False
-       self._cmake.definitions["ZSTD_BUILD_STATIC"] = not self.options.shared
-       self._cmake.definitions["ZSTD_BUILD_SHARED"] = self.options.shared
-       self._cmake.definitions["ZSTD_MULTITHREAD_SUPPORT"] = self.options.threading
-       if tools.Version(self.version) < "1.4.3":
-           self._cmake.definitions["CMAKE_POLICY_DEFAULT_CMP0042"] = "NEW"
-       self._cmake.configure(build_folder=self._build_subfolder)
-       self._cmake.build()

+   def generate(self):
+       tc = CMakeToolchain(self)
+       tc.variables["ZSTD_BUILD_PROGRAMS"] = self.options.build_programs
+       tc.variables["ZSTD_BUILD_STATIC"] = not self.options.shared or self.options.build_programs
+       tc.variables["ZSTD_BUILD_SHARED"] = self.options.shared
+       tc.variables["ZSTD_MULTITHREAD_SUPPORT"] = self.options.threading
+       if Version(self.version) < "1.5.6":
+           tc.cache_variables["CMAKE_POLICY_DEFAULT_CMP0042"] = "NEW”
+       tc.generate()
```

The Conan Way – a zstd diff (4/5)

```
    def package(self):
-       self.copy(pattern="LICENSE", dst="licenses", src=self._source_subfolder)
-       cmake = self._configure_cmake()
+       copy(self, "LICENSE", src=self.source_folder, dst=os.path.join(self.package_folder, "licenses"))
+       cmake = CMake(self)
        cmake.install()
-       tools.rmdir(os.path.join(self.package_folder, "lib", "cmake"))
-       tools.rmdir(os.path.join(self.package_folder, "lib", "pkgconfig"))
+       rmdir(self, os.path.join(self.package_folder, "lib", "cmake"))
+       rmdir(self, os.path.join(self.package_folder, "lib", "pkgconfig"))
+       rmdir(self, os.path.join(self.package_folder, "share"))
+
+       if self.options.shared and self.options.build_programs:
+           rm(self, "*_static.*", os.path.join(self.package_folder, "lib"))
+           for lib in glob.glob(os.path.join(self.package_folder, "lib", "*.a")):
+               if not lib.endswith(".dll.a"):
+                   os.remove(lib)
```

The Conan Way – a zstd diff (5/5)

```
    def package_info(self):
        zstd_cmake = "libzstd_shared" if self.options.shared else "libzstd_static"
        self.cpp_info.set_property("cmake_file_name", "zstd")
-       self.cpp_info.set_property("cmake_target_name", "zstd::{}".format(zstd_cmake))
+        self.cpp_info.set_property("cmake_target_name", f"zstd::{zstd_cmake}")
        self.cpp_info.set_property("pkg_config_name", "libzstd")
-       self.cpp_info.components["zstdlib"].set_property("pkg_config_name", "libzstd")
-       self.cpp_info.components["zstdlib"].names["cmake_find_package"] = zstd_cmake
-       self.cpp_info.components["zstdlib"].names["cmake_find_package_multi"] = zstd_cmake
-       self.cpp_info.components["zstdlib"].set_property("cmake_target_name", "zstd::{}".format(zstd_cmake))
-       self.cpp_info.components["zstdlib"].libs = tools.collect_libs(self)
+       self.cpp_info.components["zstdlib"].libs = collect_libs(self)
        if self.settings.os in ["Linux", "FreeBSD"]:
            self.cpp_info.components["zstdlib"].system_libs.append("pthread")
+       # TODO: Remove after dropping Conan 1.x from ConanCenterIndex
+       self.cpp_info.components["zstdlib"].set_property("cmake_target_name", f"zstd::{zstd_cmake}")
+       self.cpp_info.components["zstdlib"].set_property("pkg_config_name", "libzstd")
```

Troubleshooting

- Non-Conan-ical way
- November 2024 – the switch
  - Where are new packages? 🤔
  - Default remote conancenter
  - Switch https://center.conan.io to https://center2.conan.io
  - Only Conan 2 packages now
  - Check changelogs, subscribe to newsletters

Troubleshooting

- Non-Conan-ical way
- November 2024 – the switch
  - Where are new packages? 🤔
  - Default remote conancenter
  - Switch https://center.conan.io to https://center2.conan.io
  - Only Conan 2 packages now
  - Check changelogs, subscribe to newsletters
- Supporting legacy
  - Conan-center V1 still exists (read-only)
  - Modifiable or custom recipes – reupload to your server or Artifactory
  - Migrate packages from leaves to the root

Troubleshooting

- Non-Conan-ical way
- November 2024 – the switch
  - Where are new packages? 🤔
  - Default remote conancenter
  - Switch https://center.conan.io to https://center2.conan.io
  - Only Conan 2 packages now
  - Check changelogs, subscribe to newsletters
- Supporting legacy
  - Conan-center V1 still exists (read-only)
  - Modifiable or custom recipes – reupload to your server or Artifactory
  - Migrate packages from leaves to the root
- Good to have your own recipe tree

Conclusion

- This talk is right there –>

- A simple library recipe migration — **zstd**
  - Before, After and Diff
- A complex example — **DuckStax Otterbrix**
  - Consumer consumer (application) and libraries
  - Own recipe tree
  - https://github.com/duckstax/conan-duckstax/blob/master/recipes/otterbrix/all/conanfile.py
- Useful links:
  - https://docs.conan.io/1/conan_v2.html
  - https://blog.conan.io/2023/06/07/New-Cheat-Sheet-For-Conan-2.html
- Contribute your favorite recipe to Conan Center Index!