CONAN

CppCon, Sept 2016

Seattle, WA, USA

CONAN
C/C++ package manager

- FOSS (MIT), including in-house server
- Decentralized/distributed, git-like
- Build system agnostic: Generators for VS, Xcode, CMake, qmake…
- Handles from source/binaries
- No lock-in

timer.cpp

```cpp
#include "Poco/Timer.h"
#include "Poco/Thread.h"
#include "Poco/Stopwatch.h"

#include <boost/regex.hpp>
#include <string>
#include <iostream>

using Poco::Timer;
using Poco::TimerCallback;
using Poco::Thread;
using Poco::Stopwatch;

class TimerExample{
public:
    TimerExample(){ _sw.start();}

    void onTimer(Timer& timer){
        std::cout << "Callback called after " << _sw.elapsed()/1000 << " milliseconds." << std::endl;
    }
private:
    Stopwatch _sw;
};

int main(int argc, char** argv){
    TimerExample example;
    Timer timer(250, 500);
    timer.start(TimerCallback<TimerExample>(example, &TimerExample::onTimer));

    Thread::sleep(3000);
    timer.stop();

    std::string s = "correct@email.com", s2="bademail";
    boost::regex expr{"\\b[a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}\\b"};
    std::cout << std::boolalpha << boost::regex_match(s, expr) << '\n';
    std::cout << std::boolalpha << boost::regex_match(s2, expr) << '\n';

    return 0;
}
```

conanfile.txt

```
[requires]
Poco/1.7.2@lasote/stable
Boost/1.60.0@lasote/stable

[generators]
cmake

[options]
Boost:shared=False
Poco:shared=False
```

Installing dependencies

$ mkdir .conan && cd .conan

$ conan install ..

//inspect conanbuildinfo.cmake

$ conan search

$ conan info ..

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project(four_c)

include(.conan/conanbuildinfo.cmake)
conan_basic_setup()

add_compile_options(-std=c++11)
add_executable(timer timer.cpp)
target_link_libraries(timer ${CONAN_LIBS})
```

Building & running

$ cmake .. -G “Visual Studio 14 Win64”

$ cmake --build . --config Release

$ bin/timer

$ *Callback called after 262 millis…*

Another configuration

$ rm –rf *

$ conan install .. –s arch=x86

$ cmake .. -G “Visual Studio 14” //wo Win64

$ cmake --build . --config Release

$ bin/timer

$ *Callback called after 262 millis…*

From sources

$ rm –rf *

$ conan install .. --build

$ cmake .. -G “Visual Studio 14”

$ cmake --build . --config Release

$ bin/timer

$ *Callback called after 262 millis…*

* Requires perl installed (I use cmder) & I had to run
“"%vs140comntools%../../VC/vcvarsall.bat", OpenSSL build needs it (to be improved)

Other generators

```
[requires]
Poco/1.7.2@lasote/stable
Boost/1.60.0@lasote/stable

[generators]
visual_studio
ycm

[options]
Boost:shared=False
Poco:shared=False
```

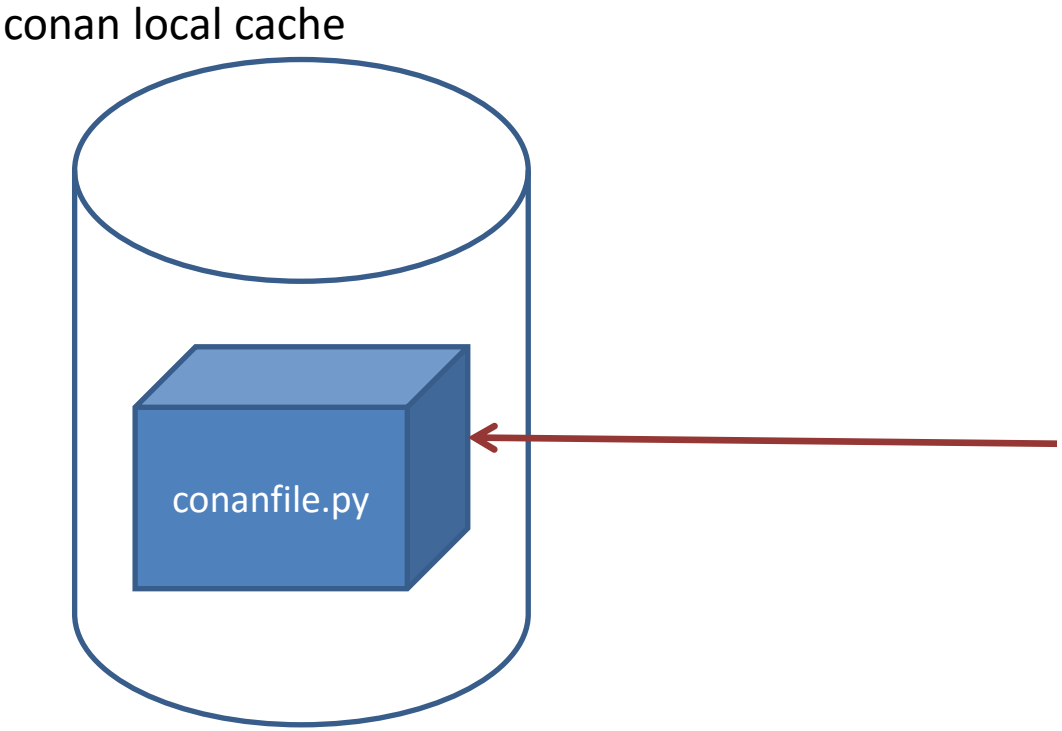
Creating packages 101

Package recipe

```python
from conans import ConanFile, CMake

class HelloConan(ConanFile):
    name = "Hello"
    version = "0.1"
    license="MIT"
    settings = "os", "compiler", "build_type", "arch"
    url = "https://github.com/memsharded/conan-hello.git"

    def source(self):
        self.run("git clone https://github.com/memsharded/hello.git")

    def build(self):
        cmake = CMake(self.settings)
        self.run('cd hello && cmake . %s' % cmake.command_line)
        self.run("cd hello && cmake --build . %s" % cmake.build_config)

    def package(self):
        self.copy("*.h", dst="include", src="hello")
        self.copy("*.lib", dst="lib", src="hello/lib")
        self.copy("*.a", dst="lib", src="hello/lib")

    def package_info(self):
        self.cpp_info.libs = ["hello"]
```

Export the package recipe

$ conan export user/testing




```
from conans import ConanFile,

class HelloConan(ConanFile):
    name = "Hello"
    version = "0.1"
    license="MIT"
    settings = "os", "compiler"
    url = "https://github.com/m
```

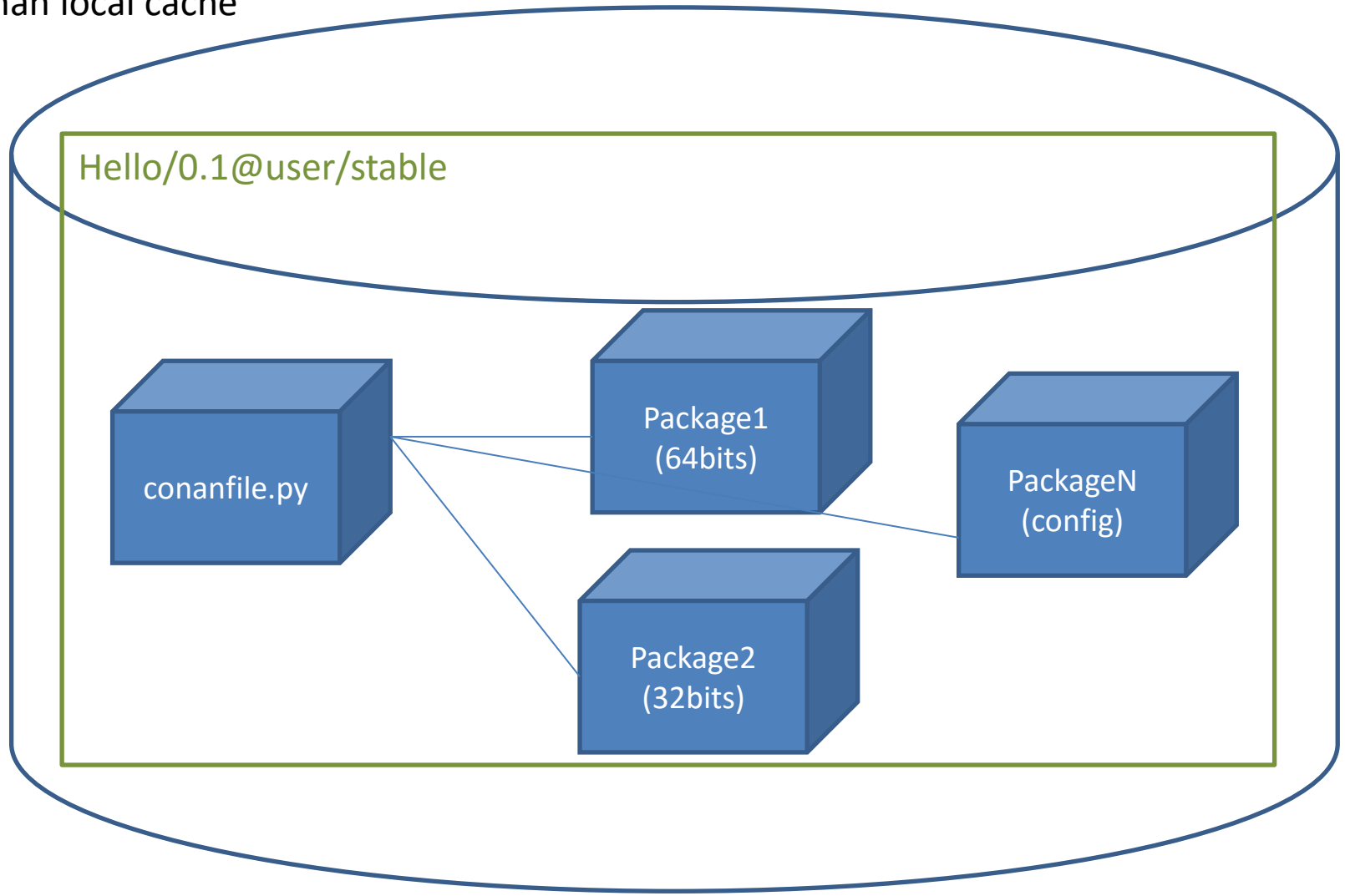
Consume the Hello package recipe

```
[requires]
Hello/0.1@user/testing

[generators]
cmake
```

```
$ conan install --build
$ cmake …
$ conan install –s arch=x86 --build
$ cmake …
```

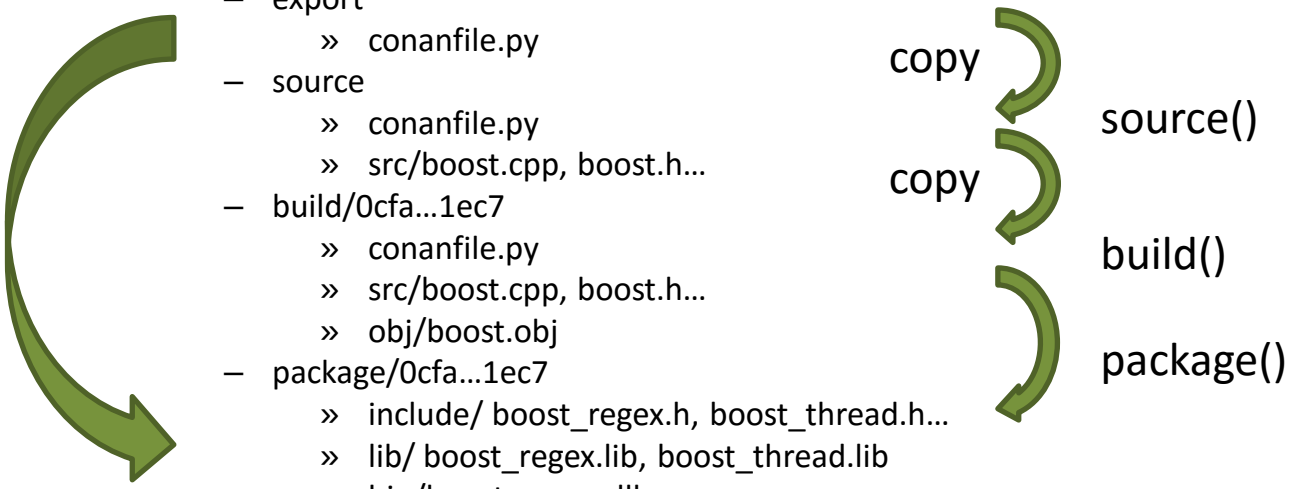
Recipe <-> Binaries

conan local cache



The local conan cache

- C:/Users/username/.conan ($CONAN_USER_HOME)
  - conan.conf
  - settings.yml
  - data
    - Boost/1.60/lasote/stable
      - export
        - » conanfile.py
      - source
        - » conanfile.py
        - » src/boost.cpp, boost.h…
      - build/0cfa…1ec7
        - » conanfile.py
        - » src/boost.cpp, boost.h…
        - » obj/boost.obj
      - package/0cfa…1ec7
        - » include/ boost_regex.h, boost_thread.h…
        - » lib/ boost_regex.lib, boost_thread.lib
        - » bin/boost_regex.dll,…
      - package/ab8d…d45e
    - zmq/
      - 1.0/memsharded/testing
      - 1.1/memsharded/



Package recipe & package binaries

Hello/0.1@user/testing: Package Recipe

- *Hello/0.1@user/testing:ID0 Package binary*
- *Hello/0.1@user/testing:ID1 Package binary*
- *Hello/0.1@user/testing:ID2 Package binary*

Package meta-information

```
class PocoConan(ConanFile):
    name = "Poco"
    version = "1.7.2"
    url="http://github.com/lasote/conan-poco"
    exports = "CMakeLists.txt"
    generators = "cmake", "txt"
    settings = "os", "arch", "compiler", "build_type"
    options = {"shared": [True, False]}
    default_options = 'shared=False'
```

Predefined settings values

```
os: [Windows, Linux, Macos, Android, iOS]
arch: [x86, x86_64, armv6, armv7, armv7hf, armv8]
compiler:
    gcc:
        version: ["4.4", "4.5", "4.6", "4.7", "4.8", "4.9",
                  "5.1", "5.2", "5.3"]
        libcxx: [libstdc++, libstdc++11]
    Visual Studio:
        runtime: [MD, MT, MTd, MDd]
        version: ["8", "9", "10", "11", "12", "14"]
    clang:
        version: ["3.3", "3.4", "3.5", "3.6", "3.7", "3.8"]
        libcxx: [libstdc++, libstdc++11, libc++]
    apple-clang:
        version: ["5.0", "5.1", "6.0", "6.1", "7.0", "7.3"]
        libcxx: [libstdc++, libc++]
build_type: [None, Debug, Release]
```

Why libcxx a setting?

- **libcxx:** [libstdc++, libstdc++11]
  - Ubuntu 14, gcc 4.9 => libstdc++
  - Ubuntu 16, gcc 5.3 => libstdc++11
  - Ubuntu 14, gcc 5.3 => ?
- What for C projects?

```
def configure(self):
    del self.settings.compiler.libcxx
```

ABI compatibility

- Major compiler versions generate different binaries:

```
compiler:
    Visual Studio:
        runtime: [MD, MT, MTd, MDd]
        version: ["8", "9", "10", "11", "12", "14"]
```

- What if not (pure C project ABI compatible):

```
def conan_info(self):
     self.info.settings.compiler.version=“ANY”
```

Settings => Build and generators

```
if(CONAN_LIBCXX STREQUAL "libstdc++11")
    add_definitions(-D_GLIBCXX_USE_CXX11_ABI=1)
elseif(CONAN_LIBCXX STREQUAL "libstdc++")
    add_definitions(-D_GLIBCXX_USE_CXX11_ABI=0)
endif()
```

Binary Package ID

```
os: Windows
arch: x86_64
compiler: gcc
compiler.version: "4.9",
compiler.libcxx: libstdc++
build_type: Release
```

settings

```
shared: True
ssl: False
```

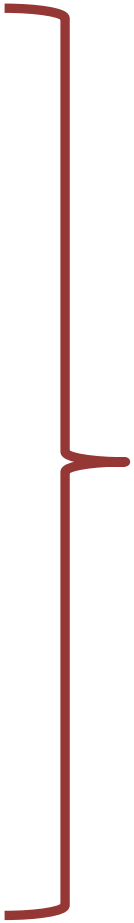
options

```
Zlib/1.Y
OpenSSL/2.Y
```

requires

SHA1

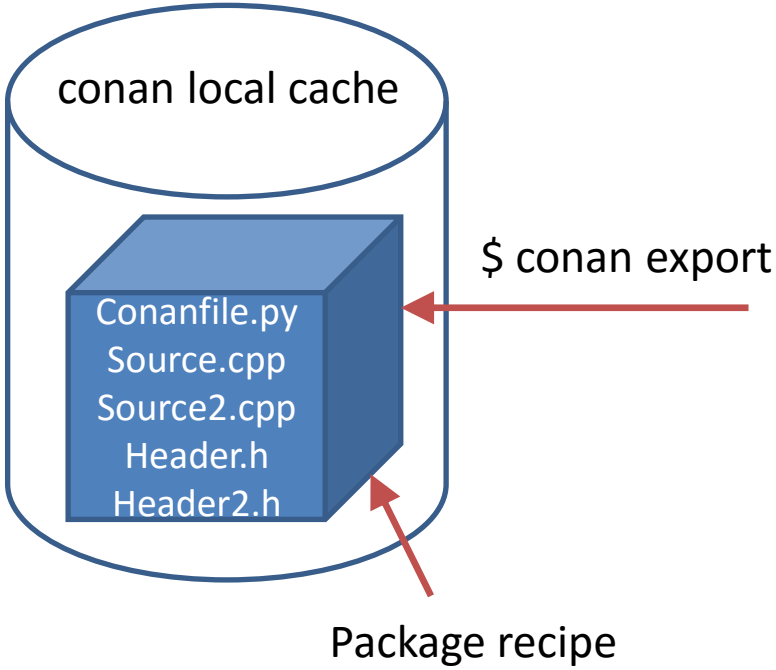
fce0123…ab56



Source origins

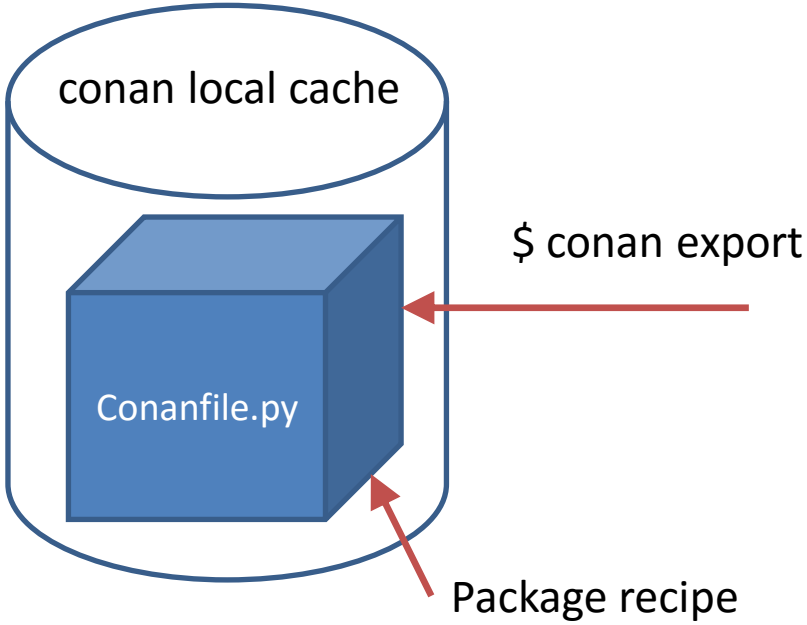
In-source

- conanfile.py inside the library repository
- exports =“src/*”, “include*”



Out-source

- conanfile.py in a different repository
- Use method source()



source()

```
class PocoConan(ConanFile):
    name = "Poco"
    version = "1.7.2“
    …


    def source(self):
        zip_name = "poco-%s-release.zip" % self.version
        download("https://github.com/pocoproject/poco/archive/%s"
                 % zip_name, zip_name)
        unzip(zip_name)
        shutil.move("poco-poco-%s-release" % self.version, "poco")
        os.unlink(zip_name)
        shutil.move("poco/CMakeLists.txt", "poco/CMakeListsOriginal.cmake")
        shutil.move("CMakeLists.txt", "poco/CMakeLists.txt")
```

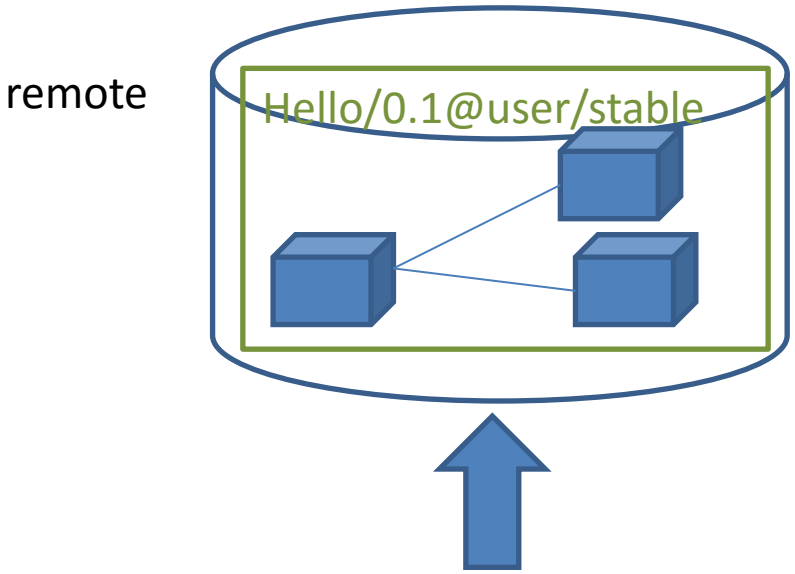
config() & build()

```
def config(self):
    if self.options.enable_netssl or self.options.force_openssl:
        self.requires.add("OpenSSL/1.0.2g@lasote/stable")
        self.options["OpenSSL"].shared = self.options.shared
    else:
        if "OpenSSL" in self.requires:
            del self.requires["OpenSSL"]

def build(self):
    cmake = CMake(self.settings)
    if self.settings.os == "Windows":
        if self.settings.compiler.runtime == "MT“:
            options_poco += " -DPOCO_MT=ON"
        else:
            options_poco += " -DPOCO_MT=OFF"

    self.run('cd poco && cmake . %s -D%s' % (cmake.command_line, options_poco))
    self.run("cd poco && cmake --build . %s" % cmake.build_config)
```

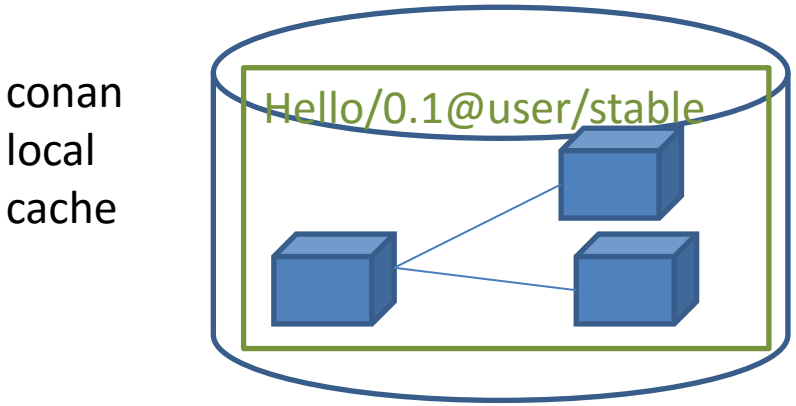
package_info()

```
def package_info(self):
    …

    if self.settings.build_type == "Debug":
        self.cpp_info.libs = ["%sd" % lib for lib in self.cpp_info.libs]

    # in linux we need to link also with these libs
    if self.settings.os == "Linux":
        self.cpp_info.libs.extend(["pthread", "dl", "rt"])

    if not self.options.shared:
        self.cpp_info.defines.extend(["POCO_STATIC=ON",
                                      "POCO_NO_AUTOMATIC_LIBS"])
        if self.settings.compiler == "Visual Studio":
            self.cpp_info.libs.extend(["ws2_32", "Iphlpapi.lib"])
```

Upload to remote

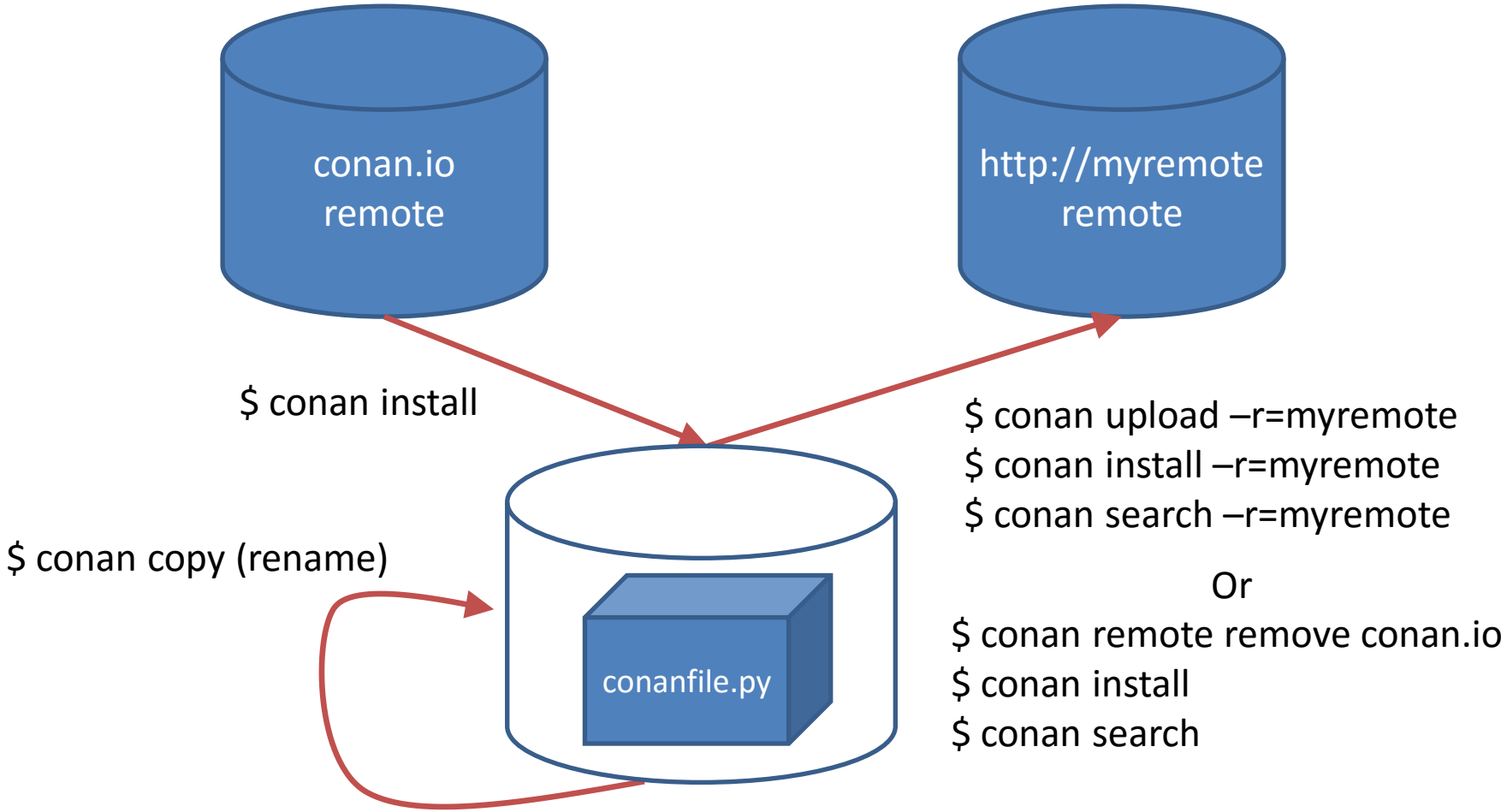


$ conan upload Hello/0.1@user/stable -r=MyRemote



Decentralized: remotes

- Multi remote, git-like



Other

- Deps:
  - Transitive and conditional dependencies
  - Conflict resolution (both for versions, channels, etc & options)
  - Private dependencies
  - Dev dependencies (installed by scopes)
- Installer packages
  - conan install cmake_installer/0.1@lasote/testing
  - activate (virtualenv)
- conan-package-tools
  - pip install conan-package-tools
  - Travis-ci (Linux, OSX), with docker
  - Appveyor (Win)
- Golang
- Rust

Thank you!

CONAN
C/C++ package manager

https://conan.io
@conan_io