Installing the mongocxx driver

Prerequisites

- Any standard Unix platform, or Windows 7 SP1+
- A compiler that suports C++11 (gcc, clang, or Visual Studio)
- CMake 3.2 or later
- boost headers (optional)

We currently test the driver with the following configurations:

	Linux	macOS	Windows
clang	3.8	7.0 (Apple)	-
gcc	4.8, 5.4	-	-
VS	n/a	n/a	14 (2015) Update 3
boost	-	1.5.3	1.6.0

Versions older the the ones listed may not work and are not supported; use them at your own risk.

Versions newer than the ones listed above should work; if you have problems, please file a bug report via JIRA (https://jira.mongodb.org/browse/CXX/).

Installation

Step 1: Install the latest version of the MongoDB C driver.

The mongocxx driver builds on top of the MongoDB C driver.

- For mongocxx-3.4.x, libmongoc 1.13.0 or later is required.
- For mongocxx-3.3.x, libmongoc 1.10.1 or later is required.
- For mongocxx-3.2.x, libmongoc 1.9.2 or later is required.
- For mongocxx-3.1.4+, libmongoc 1.7.0 or later is required.
- For mongocxx-3.1.[0-3], libmongoc 1.5.0 or later is required.
- For mongocxx-3.0.x, we recommend the last 1.4.x version of libmongoc

Unless you know that your package manager offers a high-enough version, you will need to download and build from the source code. Get a tarball from the C Driver releases (https://github.com/mongodb/mongo-c-driver/releases) page.

Follow the instructions for building from a tarball at Installing libmongoc (http://mongoc.org/libmongoc/current/installing.html).

Industry best practices and some regulations require the use of TLS 1.1 or newer. The MongoDB C Driver supports TLS 1.1 on Linux if OpenSSL is at least version 1.0.1. On macOS and Windows, the C Driver uses native TLS implementations that support TLS 1.1.

Step 2: Choose a C++17 polyfill

The mongocxx driver uses the C++17 features `std::optional` and `std::string_view`. To compile the mongocxx driver for pre-C++17, you must choose one of the following implementations for these features:

MNMLSTC/core (*default for non-Windows platforms*) Select with `-DBSONCXX_POLY_USE_MNMLSTC=1`. **NOTE**: This option vendors a header-only installation of MNMLSTC/core into the bsoncxx library installation and will therefore download MLNMLSTC from GitHub during the build process. If you already have an available version of MNMLSTC on your system, you can avoid the download step by using `-DBSONCXX_POLY_USE_SYSTEM_MNMLSTC`.

Boost (*default for Windows platforms*) Select with `-DBSONCXX_POLY_USE_BOOST=1`. This is currently the only option if you are using MSVC.

`std::experimental` Select with `-DBSONCXX_POLY_USE_STD_EXPERIMENTAL=1`. If your toolchain's standard library provides `optional` and `string_view` in the namespace `std::experimental`, you can use this option. Be aware that your standard library's `std::experimental` implementation may change over time, breaking binary compatibility in unexpected ways. Note that this polyfill is *not* recommended and is unsupported.

Most users should be fine sticking with the default. However, if you have an existing application which makes heavy use of one of the available libraries, you may prefer to build the mongocxx driver against the same library.

DO NOT change your project's polyfill if you need to create a stable binary interface.

Step 3: Download the latest version of the mongocxx driver.

To get the source via git, the `releases/stable` branch will track the most recent stable release. For example, to work from a shallow checkout of the stable release branch:

```
git clone https://github.com/mongodb/mongo-cxx-driver.git \
    --branch releases/stable --depth 1
cd mongo-cxx-driver/build
```

If you prefer to download a tarball, look on the mongocxx releases (https://github.com/mongodb/mongo-cxx-driver/releases) page for a link to the release tarball for the version you wish you install. For example, to download version 3.3.1:

```
curl -OL https://github.com/mongodb/mongo-cxx-driver/archive/r3.3.1.tar.gz
tar -xzf r3.3.1.tar.gz
cd mongo-cxx-driver-r3.3.1/build
```

Make sure you change to the `build` directory of whatever source tree you obtain.

Step 4: Configure the driver

On Unix systems, `libmongoc` installs into `/usr/local` by default. To configure `mongocxx` for installation into `/usr/local` as well, use the following `cmake` command:

(***NOTE***: The trailing `..` below is important! Don't omit it.)

```
cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr/local ..
```

`/usr/local` may be replaced with the directory that `libmongoc` was installed to.

If you need to install mongocxx into a different directory than the directory where libmongoc is installed, you must instruct cmake on how to find the libmongoc installation directory. The procedure for doing so varies between mongocxx versions:

- Users building mongocxx versions 3.2.x or newer should use CMAKE_PREFIX_PATH to specify the libmongoc installation directory. For example (make sure to replace /your/cdriver/prefix and /your/cxxdriver/prefix with the prefix used for installing the C driver and the desired installation prefix, respectively):

```
cmake -DCMAKE_BUILD_TYPE=Release \
    -DCMAKE_INSTALL_PREFIX=/your/cxxdriver/prefix \
    -DCMAKE_PREFIX_PATH=/your/cdriver/prefix ..
```

- *Note*: If you need multiple paths in CMAKE_PREFIX_PATH , separate them with a semicolon like this: -DCMAKE_PREFIX_PATH="/your/cdriver/prefix;/some/other/path"
- Users building mongocxx versions 3.1.x and 3.0.x should specify the libmongoc installation directory by using the -DLIBMONGOC_DIR and -DLIBBSON_DIR options to cmake . See the following example, which assumes that both libmongoc and libbson are installed into /your/cdriver/prefix :

```
cmake -DCMAKE_BUILD_TYPE=Release \
    -DCMAKE_INSTALL_PREFIX=/your/cxxdriver/prefix
    -DLIBMONGOC_DIR=/your/cdriver/prefix
    -DLIBBSON_DIR=/your/cdriver/prefix ..
```

Remember: to select a polyfill, pass the option to cmake . For example, to select the Boost polyfill, substitute the cmake line with the following:

```
cmake -DCMAKE_BUILD_TYPE=Release -DBSONCXX_POLY_USE_BOOST=1 \
    -DCMAKE_INSTALL_PREFIX=/usr/local ..
```

On Windows, here's an example of how to configure for MSVC (assuming libmongoc and libbson are in C:\mongo-c-driver as given in the mongoc Windows installation instructions (http://mongoc.org/libmongoc/current/installing.html#building-windows) and boost is in

```
'C:\Program Files (x86)\CMake\bin\cmake.exe' -G "Visual Studio 14 2015 Win64"
    -DCMAKE_INSTALL_PREFIX=C:\mongo-cxx-driver
    -DCMAKE_PREFIX_PATH=C:\mongo-c-driver
    -DBOOST_ROOT=C:\local\boost_1_59_0 ..
```

mongocxx builds shared libraries by default. This is the recommended build setting for novice users.

- Note for users of mongocxx 3.2.x and newer: advanced users may build static libraries, if desired, by specifying `-DBUILD_SHARED_LIBS=OFF` to CMake. Specifying this option will introduce a dependency on the `libmongoc` static libraries. Linking an application against both shared `libmongoc` and static `mongocxx` is not supported, nor is linking against both static `libmongoc` and shared `mongocxx`.

For building with Visual Studio 2017 (without a C++17 polyfill), it is necessary to configure with an additional option, `/Zc:__cplusplus` to opt into the correct definition of `__cplusplus` (problem described here (https://blogs.msdn.microsoft.com/vcblog/2018/04/09/msvc-now-correctly-reports-__cplusplus/)):

```
'C:\Program Files (x86)\CMake\bin\cmake.exe' -G "Visual Studio 15 2017 Win64"
    -DCMAKE_INSTALL_PREFIX=C:\mongo-cxx-driver
    -DCMAKE_PREFIX_PATH=C:\mongo-c-driver
    -DCMAKE_CXX_STANDARD=17
    -DCMAKE_CXX_FLAGS="/Zc:__cplusplus" ..
```

Step 5: Build and install the driver

If you are using the default MNMLSTC polyfill and are installing to a directory requiring root permissions, you should install the polyfill with `sudo` before running `make` so you don't have to run all of `make` with `sudo`:

```
# Only for MNMLSTC polyfill
sudo make EP_mnmlstc_core
```

Once MNMLSTC is installed, or if you are using a different polyfill, build and install the driver:

```
make && sudo make install
```

On Windows, build and install from the command line like this:

```
msbuild.exe ALL_BUILD.vcxproj
msbuild.exe INSTALL.vcxproj
```

Step 6: Test your installation

Save the following source file with the filename `test.cpp` underneath any directory:

```
#include <iostream>

#include <bsoncxx/builder/stream/document.hpp>
#include <bsoncxx/json.hpp>

#include <mongocxx/client.hpp>
#include <mongocxx/instance.hpp>

int main(int, char**) {
    mongocxx::instance inst{};
    mongocxx::client conn{mongocxx::uri{}};

    bsoncxx::builder::stream::document document{};

    auto collection = conn["testdb"]["testcollection"];
    document << "hello" << "world";

    collection.insert_one(document.view());
    auto cursor = collection.find({});

    for (auto&& doc : cursor) {
        std::cout << bsoncxx::to_json(doc) << std::endl;
    }
}
```

Compiling with the help of CMake

If you are using CMake for your project, you can use the find_package() directive to enable compiling and linking against `mongocxx`. The find_package() directive will set variables in the current environment (e.g. `LIBMONGOCXX_INCLUDE_DIRS`, `LIBMONGOCXX_LIBRARIES`, etc.) that need to be propagated to your build targets. If you have installed `mongocxx` or `libmongoc` to a non-standard location on your system, you will need to set `CMAKE_PREFIX_PATH` to the library installation prefix (specified at build time with `CMAKE_INSTALL_PREFIX`) when running `cmake`.

In the `mongocxx` source repository (versions 3.2.x or newer only), see the directory `examples/projects/mongocxx/cmake` for an example CMake application which uses the shared library (the default option), and an example CMake application which uses the static library (advanced users only).

Compiling with the help of pkg-config

Compile the test program above with the following command:

```
c++ --std=c++11 test.cpp -o test $(pkg-config --cflags --libs libmongocxx)
```

Advanced users who are using the static library must replace the `libmongocxx` argument to `pkg-config` above with `libmongocxx-static` (this requires mongocxx 3.2.x or newer).

If you installed to somewhere not in your pkg-config search path, remember to set the `PKG_CONFIG_PATH` environment variable first:

```
export PKG_CONFIG_PATH="$MY_INSTALL_PREFIX/lib/pkgconfig"
```

Compiling without pkg-config or CMake

If you aren't using CMake for your project and you don't have pkg-config available, you will need to set include and library flags manually on the command line or in your IDE.

Here's an example expansion of the compilation line above, on a system where mongocxx and libmongoc are installed in `/usr/local`:

```
c++ --std=c++11 test.cpp -o test \
    -I/usr/local/include/mongocxx/v_noabi \
    -I/usr/local/include/bsoncxx/v_noabi \
    -L/usr/local/lib -lmongocxx -lbsoncxx
```

Advanced users only: here is an example expansion on the same system of the compilation line above when static libraries are being used. Note that the preprocessor defines `MONGOCXX_STATIC` and `BSONCXX_STATIC` must be defined in all source files that include mongocxx headers; failure to do so will result in difficult-to-diagnose linker errors.

```
c++ --std=c++11 test.cpp -o test \
    -DMONGOCXX_STATIC -DBSONCXX_STATIC -DMONGOC_STATIC -DBSON_STATIC \
    -I/usr/local/include/libmongoc-1.0 \
    -I/usr/local/include/libbson-1.0 \
    -I/usr/local/include/mongocxx/v_noabi \
    -I/usr/local/include/bsoncxx/v_noabi \
    -L/usr/local/lib -lmongocxx-static -lbsoncxx-static
    -lmongoc-static-1.0 -lsasl2 -lssl -lcrypto -lbson-static-1.0 -lm -lpthread
```

Compiling with MSVC

To compile on MSVC, you will need to setup your project to include all the necessary include paths, library paths, preprocessor defines, and link libraries. To do this, you can set these values either by the UI or by editing the XML `.vcxproj` file directly. To confirm you have everything setup correctly, here are the `PropertyGroup` and `ItemDefinitionGroup` settings for a Debug x64 build as an example:

```
<PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
  <LinkIncremental>true</LinkIncremental>
  <IncludePath>c:\local\boost_1_59_0\;C:\mongo-cxx-driver\include\mongocxx\v_noabi;C:\mongo
  <LibraryPath>c:\mongo-c-driver\lib\;c:\mongo-cxx-driver\lib\;$(LibraryPath)</LibraryPath>
</PropertyGroup>
<ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
  <ClCompile>
    <PrecompiledHeader>Use</PrecompiledHeader>
    <WarningLevel>Level3</WarningLevel>
    <Optimization>Disabled</Optimization>
    <PreprocessorDefinitions>MONGOCXX_STATIC;BSONCXX_STATIC;_DEBUG;_CONSOLE;%(PreprocessorD
    <SDLCheck>true</SDLCheck>
  </ClCompile>
  <Link>
    <SubSystem>Console</SubSystem>
    <GenerateDebugInformation>true</GenerateDebugInformation>
    <AdditionalDependencies>libmongocxx.lib;libbsoncxx.lib;mongoc-static-1.0.lib;bson-1.0.l
  </Link>
</ItemDefinitionGroup>
```

Configuring the mongocxx driver ➞ (/mongocxx-v3/configuration/)