Other languages: 繁體中文

Installation

This section takes Ubuntu 24.04, CentOS 7.5, MacOS 12.2 as an example to introduce the installation process. Other systems are similar;

System Requirements

- The Linux kernel should be not lower than 2.6.9, 64-bit version;
- The gcc version is not less than 5.4.0, suggest to use version 11 or above;
- Use cmake as the build tool, and the cmake version should be not less than 3.5;
- Use git as the version management tool;

Library Dependencies

- Built-in
  - trantor, a non-blocking I/O C++ network library, also developed by the author of Drogon, has been used as a git repository submodule, no need to install in advance;
- Mandatory
  - jsoncpp, JSON's c++ library, the version should be **no less than 1.7**;
  - libuuid, generating c library of uuid;
  - zlib, used to support compressed transmission;
- Optional
  - boost, the version should be **no less than 1.61**, is required only if the C++ compiler does not support C++ 17 and if the STL doesn't fully support `std::filesystem`.
  - OpenSSL, after installed, drogon will support HTTPS as well, otherwise drogon only supports HTTP.
  - c-ares, after installed, drogon will be more efficient with DNS;
  - libbrotli, after installed, drogon will support brotli compression when sending HTTP responses;
  - the client development libraries of postgreSQL, mariadb and sqlite3, if one or more of them is installed, drogon will support access to the according database.
  - hiredis, after installed, drogon will support access to redis.
  - gtest, after installed, the unit tests can be compiled.
  - yaml-cpp, after installed, drogon will support config file with yaml format.

System Preparation Examples

Ubuntu 24.04

- Environment

```
sudo apt install git gcc g++ cmake
```

- jsoncpp

```
sudo apt install libjsoncpp-dev
```

- uuid

```
sudo apt install uuid-dev
```

- zlib

```
sudo apt install zlib1g-dev
```

- OpenSSL (Optional, if you want to support HTTPS)

```
sudo apt install openssl libssl-dev
```

Arch Linux

- Environment

```
sudo pacman -S git gcc make cmake
```

- jsoncpp

```
sudo pacman -S jsoncpp
```

- uuid

```
sudo pacman -S uuid
```

- zlib

```
sudo pacman -S zlib
```

- OpenSSL (Optional, if you want to support HTTPS)

```
sudo pacman -S openssl libssl
```

CentOS 7.5

- Environment

```
yum install git
yum install gcc
yum install gcc-c++
```

```
# The default installed cmake version is too low, use source
installation
git clone https://github.com/Kitware/CMake
cd CMake/
./bootstrap && make && make install
```

```
# Upgrade gcc
yum install centos-release-scl
yum install devtoolset-11
scl enable devtoolset-11 bash
```

> **Note: Command `scl enable devtoolset-11 bash` only activate the new gcc temporarily until the session is end. If you want to always use the new gcc, you could run command `echo "scl enable devtoolset-11 bash" >> ~/.bash_profile`, system will automatically activate the new gcc after restarting.**

- jsoncpp

```
git clone https://github.com/open-source-parsers/jsoncpp
cd jsoncpp/
mkdir build
cd build
cmake ..
make && make install
```

- uuid

```
yum install libuuid-devel
```

- zlib

```
yum install zlib-devel
```

- OpenSSL (Optional, if you want to support HTTPS)

```
yum install openssl-devel
```

MacOS 12.2

- Environment

  All the essentials are inherent in MacOS, you only need to upgrade it.

```
# upgrade gcc
brew upgrade
```

- jsoncpp

```
brew install jsoncpp
```

- uuid

```
brew install ossp-uuid
```

- zlib

```
yum install zlib-devel
```

- OpenSSL (Optional, if you want to support HTTPS)

```
brew install openssl
```

Windows

- Environment (Visual Studio 2019) Install Visual Studio 2019 professional 2019, at least included these options:
  - MSVC C++ building tools
  - Windows 10 SDK
  - C++ CMake tools for windows
  - Test adaptor for Google Test

conan package manager could provide all dependencies that Drogon projector needs。If python is installed on system, you could install conan package manager via pip.

```
pip install conan
```

> of course you can download the installation file from conan official website to install it also.

Createconanfile.txtand add the following content to it:

- jsoncpp

  ```
  [requires]
  jsoncpp/1.9.4
  ```

- uuid

  No installation is required, the Windows 10 SDK already includes the uuid library.

- zlib

  ```
  [requires]
  zlib/1.2.11
  ```

- OpenSSL (Optional, if you want to support HTTPS)

  ```
  [requires]
  openssl/1.1.1t
  ```

Database Environment (Optional)

> **Note: These libraries below are not mandatory. You could choose to install one or more database according to your actual needs.**

> **Note: If you want to develop your webapp with database, please install the database develop environment first, then install drogon, otherwise you will encounter a `NO DATABASE FOUND` issue.**

- **PostgreSQL**

  PostgreSQL's native C library libpq needs to be installed. The installation is as follows:

  - ubuntu 16: `sudo apt-get install postgresql-server-dev-all`
  - ubuntu 18: `sudo apt-get install postgresql-all`
  - ubuntu 24: `sudo apt-get install postgresql-all`
  - arch: `sudo pacman -S postgresql`
  - centOS 7: `yum install postgresql-devel`
  - MacOS: `brew install postgresql`
  - Windows conanfile: `libpq/13.4`

- **MySQL**

  MySQL's native library does not support asynchronous read and write. Fortunately, MySQL also has a version of MariaDB maintained by the original developer community. This version is compatible with MySQL, and its development library supports asynchronous read and write. Therefore, Drogon uses the MariaDB development library to provide the right MySQL support, as a best practice，your operating system shouldn't install both Mysql and MariaDB at the same time.

  MariaDB installation is as follows：

  - ubuntu 18.04: `sudo apt install libmariadbclient-dev`
  - ubuntu 24.04: `sudo apt install libmariadb-dev-compat libmariadb-dev`
  - arch: `sudo pacman -S mariadb`
  - centOS 7: `yum install mariadb-devel`
  - MacOS: `brew install mariadb`
  - Windows conanfile: `libmariadb/3.1.13`

- **Sqlite3**

  - ubuntu: `sudo apt-get install libsqlite3-dev`
  - arch: `sudo pacman -S sqlite3`
  - centOS: `yum install sqlite-devel`
  - MacOS: `brew install sqlite3`
  - Windows conanfile: `sqlite3/3.36.0`

- **Redis**

  - ubuntu: `sudo apt-get install libhiredis-dev`
  - ubuntu: `sudo pacman -S redis`
  - centOS: `yum install hiredis-devel`
  - MacOS: `brew install hiredis`
  - Windows conanfile: `hiredis/1.0.0`

> **Note: Some of the above commands only install the development library. If you want to install a server also, please use Google search yourself.**

Drogon Installation

Assuming that the above environment and library dependencies are all ready, the installation process is very simple;

- **Install by source in Linux**

```
cd $WORK_PATH
git clone https://github.com/drogonframework/drogon
cd drogon
git submodule update --init
mkdir build
cd build
cmake ..
make && sudo make install
```

> The default is to compile the debug version. If you want to compile the release version, the cmake command should take the following parameters:

```
cmake -DCMAKE_BUILD_TYPE=Release ..
```

After the installation is complete, the following files will be installed in the system（One can change the installation location with the CMAKE_INSTALL_PREFIX option）:

  - The header file of drogon is installed into /usr/local/include/drogon;
  - The drogon library file libdrogon.a is installed into /usr/local/lib;
  - Drogon's command line tool drogon_ctl is installed into /usr/local/bin;
  - The trantor header file is installed into /usr/local/include/trantor;
  - The trantor library file libtrantor.a is installed into /usr/local/lib;

- **Install by source in Windows**

1. Download drogon source

   Open the Windows taskbar search box, search for x64 Native Tools, and select x64 Native Tools Command Prompt for VS 2019 as your command-line tool.

```
cd $WORK_PATH
git clone https://github.com/drogonframework/drogon
cd drogon
git submodule update --init
```

2. Install dependencies

   install dependencies via conan:

```
mkdir build
cd build
conan profile detect --force
conan install .. -s compiler="msvc" -s compiler.version=193  -s
compiler.cppstd=17 -s build_type=Debug  --output-folder . --
build=missing
```

> Modify `conanfile.txt` to change the version of dependencies.

3. Compile and install

```
cmake ..  -DCMAKE_BUILD_TYPE=Debug -
DCMAKE_TOOLCHAIN_FILE="conan_toolchain.cmake" -
DCMAKE_POLICY_DEFAULT_CMP0091=NEW -DCMAKE_INSTALL_PREFIX="D:"
cmake --build . --parallel --target install
```

> **Note: Must keep build type same in conan and cmake.**

After the installation is complete, the following files will be installed in the system（One can change the installation location with the `CMAKE_INSTALL_PREFIX` option）:

- The header file of drogon is installed into `D:/include/drogon`;
- The drogon library file drogon.dll is installed into `D:/bin`;
- Drogon's command line tool drogon_ctl.exe is installed into `D:/bin`;
- The trantor header file is installed into `D:/include/trantor`;
- The trantor library file trantor.dll is installed into `D:/lib`;

Add `bin` and `cmake` directory to `path`：

```
D:\bin
```

```
D:\lib\cmake\Drogon
```

```
D:\lib\cmake\Trantor
```

- **Install by vcpkg in Windows**

  Lazzy to read

  Install vcpkg:

  1. Install `vcpkg` by `git`.

```
git clone https://github.com/microsoft/vcpkg
cd vcpkg
./bootstrap-vcpkg.bat
```

> note: to update your vcpkg, you just need to type `git pull`

2. add `vpckg` to your windows environment variables ***path***.
3. Now check if vcpkg already installed properly, just type `vcpkg` or `vcpkg.exe`

Now Install Drogon:

1. To install drogon framework. Type:
   - 32-Bit: `vcpkg install drogon`
   - 64-Bit: `vcpkg install drogon:x64-windows`
   - extra : `vcpkg install jsoncpp:x64-windows zlib:x64-windows openssl:x64-windows sqlite3:x64-windows libpq:x64-windows libpqxx:x64-windows drogon[core,ctl,sqlite3,postgres,orm]:x64-windows`

   note:
   - if there's any package is/are uninstalled and you got error, just install that package. e.g.:

     zlib : `vcpkg install zlib` or `vcpkg install zlib:x64-windows` for 64-Bit
   - to check what already installed:

     `vcpkg list`
   - To use `drogon_ctl`, type `vcpkg install drogon[ctl]`(32-bit) or `vcpkg install drogon[ctl]:x64-windows`(64-bit). Type `vcpkg search drogon` for more installation feature options.
2. To add ***drogon_ctl*** command and dependencies, you need to add some variables. By following this guide, you just need to add:

```
C:\dev\vcpkg\installed\x64-windows\tools\drogon
```

```
C:\dev\vcpkg\installed\x64-windows\bin
```

```
C:\dev\vcpkg\installed\x64-windows\lib
```

```
C:\dev\vcpkg\installed\x64-windows\include
```

```
C:\dev\vcpkg\installed\x64-windows\share
```

```
C:\dev\vcpkg\installed\x64-windows\debug\bin
```

```
C:\dev\vcpkg\installed\x64-windows\debug\lib
```

to your windows ***environment variables***. Then restart/re-open your ***powershell***.

3. reload/re-open your ***powershell***, then type: `drogon_ctl` or `drogon_ctl.exe` if:

```
usage: drogon_ctl [-v | --version] [-h | --help] <command>
[<args>]
commands list:
create                  create some source files(Use 'drogon_ctl
help create' for more information)
help                    display this message
press                   Do stress testing(Use 'drogon_ctl help
press' for more information)
version                 display version of this tool
```

showed up, you are good to go.

> Note: you need to be familiar with building cpp libraries by using: independent `gcc` or `g++` (***msys2***, ***mingw-w64***, ***tdm-gcc***) or Microsoft Visual Studio compiler

> consider use ***make.exe/nmake.exe/ninja.exe*** as cmake generator since configuration and build behavior is same as _make* linux, if some devs using Linux/Windows and you are planning to deploy on Linux environment, it's less prone error when switching operating-system.

- **Use Docker Image**

  We also provide a pre-build docker image on the docker hub. All dependencies of Drogon and Drogon itself are already installed in the docker environment, where users can build Drogon-based applications directly.

- **Use Nix Package**

  There is a Nix package for Drogon which was released in version 21.11.

  > **if you haven't installed Nix:** You can follow the instructions on the NixOS website.

You can use the package by adding the following `shell.nix` to your project root:

```
{ pkgs ? import <nixpkgs> {} }:
pkgs.mkShell {
  nativeBuildInputs = with pkgs; [
    cmake
  ];

  buildInputs = with pkgs; [
    drogon
  ];

}
```

Enter the shell by running `nix-shell`. This will install Drogon and enter you into an environment with all its dependencies.

The Nix package has a few options which you can configure according to your needs:

option	default value
sqliteSupport	true
postgresSupport	false
redisSupport	false
mysqlSupport	false

Here is an example of how you can change their values:

```
  buildInputs = with pkgs; [
    (drogon.override {
      sqliteSupport = false;
    })
  ];
```

- **Use CPM.cmake**

You can use CPM.cmake to include the drogon source code:

```
include(cmake/CPM.cmake)

CPMAddPackage(
    NAME drogon
    VERSION 1.7.5
    GITHUB_REPOSITORY drogonframework/drogon
    GIT_TAG v1.7.5
```

```
)

target_link_libraries(${PROJECT_NAME} PRIVATE drogon)
```

- **Include drogon source code locally**

  Of course, you can also include the drogon source in your project. Suppose you put the drogon under the third_party of your project directory (don't forget to update submodule in the drogon source directory). Then, you only need to add the following two lines to your project's cmake file:

  ```
  add_subdirectory(third_party/drogon)
  target_link_libraries(${PROJECT_NAME} PRIVATE drogon)
  ```

Next: Quick Start