安裝

原文：ENG-02-Installation.md

本章以 Ubuntu 24.04、CentOS 7.5、MacOS 12.2 為例介紹安裝流程，其他系統大致相同。

系統需求

- Linux 核心版本需不低於 2.6.9，且為 64 位元；
- gcc 版本需不低於 5.4.0，建議使用 11 以上版本；
- 建置工具採用 cmake，cmake 版本需不低於 3.5；
- 版本管理工具採用 git；

套件相依性

- 內建
  - trantor：非阻塞 I/O C++ 網路函式庫，由 Drogon 作者開發，已作為 git 子模組，無需事先安裝；
- 必要
  - jsoncpp：C++ 的 JSON 函式庫，版本需 **不低於 1.7**；
  - libuuid：產生 uuid 的 C 函式庫；
  - zlib：支援壓縮傳輸；
- 選用
  - boost：版本需 **不低於 1.61**，僅當 C++ 編譯器不支援 C++17 或 STL 不完整支援 `std::filesystem` 時才需安裝。
  - OpenSSL：安裝後可支援 HTTPS，否則僅支援 HTTP。
  - c-ares：安裝後可提升 DNS 效率；
  - libbrotli：安裝後可支援 brotli 壓縮 HTTP 回應；
  - PostgreSQL、MariaDB、sqlite3 的開發函式庫，安裝任一即可支援對應資料庫；
  - hiredis：安裝後可支援 Redis；
  - gtest：安裝後可編譯單元測試；
  - yaml-cpp：安裝後可支援 yaml 格式設定檔。

系統準備範例

Ubuntu 24.04

- 環境

```
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（選用，若需支援 HTTPS）

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

Arch Linux

- 環境

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

- jsoncpp

```
sudo pacman -S jsoncpp
```

- uuid

```
sudo pacman -S uuid
```

- zlib

```
sudo pacman -S zlib
```

- OpenSSL（選用，若需支援 HTTPS）

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

CentOS 7.5

- 環境

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

```
# 預設 cmake 版本過低，請以原始碼安裝
git clone https://github.com/Kitware/CMake
cd CMake/
./bootstrap && make && make install
```

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

> **注意：`scl enable devtoolset-11 bash` 只會暫時啟用新版 gcc，若需永久啟用，請執行 `echo "scl enable devtoolset-11 bash" >> ~/.bash_profile`，系統重啟後自動啟用新版 gcc。**

- 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（選用，若需支援 HTTPS）

```
```

```
yum install openssl-devel
```

MacOS 12.2

- 環境

  MacOS 內建基本工具，只需升級即可。

  ```
  # 升級 gcc
  brew upgrade
  ```

- jsoncpp

  ```
  brew install jsoncpp
  ```

- uuid

  ```
  brew install ossp-uuid
  ```

- zlib

  ```
  yum install zlib-devel
  ```

- OpenSSL（選用，若需支援 HTTPS）

  ```
  brew install openssl
  ```

Windows

- 環境（Visual Studio 2019） 安裝 Visual Studio 2019 專業版，至少包含：
  - MSVC C++ 編譯工具
  - Windows 10 SDK
  - C++ CMake 工具
  - Google Test 測試介面

`conan` 套件管理工具可提供 Drogon 所需所有相依套件。若系統已安裝 python，可透過 pip 安裝 `conan`：

```
pip install conan
```

> 也可至 `conan` 官方網站下載安裝檔。

建立 `conanfile.txt` 並加入下列內容：

- jsoncpp

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

- uuid

  無需安裝，Windows 10 SDK 已內建 uuid 函式庫。

- zlib

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

- OpenSSL（選用，若需支援 HTTPS）

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

資料庫環境（選用）

> **注意：以下函式庫非必須，可依實際需求選擇安裝一種或多種資料庫。 若需開發資料庫相關 Web 應用，請先安裝資料庫開發環境再安裝 drogon，否則會遇到 `NO DATABASE FOUND` 問題。**

- **PostgreSQL**

  需安裝 PostgreSQL 原生 C 函式庫 libpq，安裝方式如下：

  - `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 原生函式庫不支援非同步讀寫，建議使用 MariaDB（由原開發社群維護，與 MySQL 相容且支援非同步），作業系統不應同時安裝 MySQL 與 MariaDB。

MariaDB 安裝方式如下：

- 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
  - arch：sudo pacman -S redis
  - centOS：yum install hiredis-devel
  - MacOS：brew install hiredis
  - Windows conanfile：hiredis/1.0.0

> **注意：部分指令僅安裝開發函式庫，若需安裝伺服器請自行查詢。**

Drogon 安裝

假設上述環境與套件皆已準備好，安裝流程如下：

- **Linux 原始碼安裝**

```
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
```

> 預設編譯為 debug 版本，若需 release 版本，cmake 指令請加下列參數：

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

安裝完成後，以下檔案會安裝至系統（可用 CMAKE_INSTALL_PREFIX 變更安裝路徑）：

- drogon 標頭檔：/usr/local/include/drogon
- drogon 函式庫檔案 libdrogon.a：/usr/local/lib
- drogon 命令列工具 drogon_ctl：/usr/local/bin
- trantor 標頭檔：/usr/local/include/trantor
- trantor 函式庫檔案 libtrantor.a：/usr/local/lib

- **Windows 原始碼安裝**

1. 下載 drogon 原始碼

   開啟 Windows 搜尋列，搜尋 x64 Native Tools，選擇 x64 Native Tools Command Prompt for VS 2019 作為命令列工具。

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

2. 安裝相依套件

   透過 `conan` 安裝：

   ```
   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
   ```

   > 修改 `conanfile.txt` 可變更套件版本。

3. 編譯與安裝

   ```
   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
   ```

> **注意：conan 與 cmake 的 build type 必須一致。**

安裝完成後，以下檔案會安裝至系統（可用 `CMAKE_INSTALL_PREFIX` 變更安裝路徑）：

- drogon 標頭檔：`D:/include/drogon`
- drogon 函式庫檔 drogon.dll：`D:/bin`

- drogon 命令列工具 drogon_ctl.exe：`D:/bin`
- trantor 標頭檔：`D:/include/trantor`
- trantor 函式庫檔 trantor.dll：`D:/lib`

新增 `bin` 與 `cmake` 目錄至 `path`：

```
D:\bin
```

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

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

- **Windows vcpkg 安裝**

  懶人包教學

  安裝 vcpkg：

  1. 以 `git` 安裝 `vcpkg`。

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

     > 更新 vcpkg 只需執行 `git pull`

  2. 將 `vcpkg` 加入 Windows 環境變數 ***path***。

  3. 檢查 vcpkg 是否安裝成功，輸入 `vcpkg` 或 `vcpkg.exe`

  安裝 Drogon：

  1. 安裝 drogon 框架，指令如下：

     - 32 位元：`vcpkg install drogon`
     - 64 位元：`vcpkg install drogon:x64-windows`
     - 進階：`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`

     注意：

     - 若有套件未安裝導致錯誤，請個別安裝。例如：

zlib：`vcpkg install zlib` 或 `vcpkg install zlib:x64-windows`（64 位元）

- 檢查已安裝套件：

  ```
  vcpkg list
  ```

- 使用 `drogon_ctl`：`vcpkg install drogon[ctl]`（32 位元）或 `vcpkg install drogon[ctl]:x64-windows`（64 位元）。輸入 `vcpkg search drogon` 可查詢更多安裝選項。

2. 新增 ***drogon_ctl*** 指令與相依套件，請將下列路徑加入環境變數：

   ```
   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
   ```

   重新啟動／開啟 ***powershell***。

3. 重新啟動／開啟 ***powershell***，輸入：`drogon_ctl` 或 `drogon_ctl.exe`，若顯示：

   ```
   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
   ```

   即安裝成功。

> 注意： 熟悉獨立 `gcc` 或 `g++`（***msys2 mingw-w64 tdm-gcc***）或 Microsoft Visual Studio 編譯器者，建議使用 ***make.exe/nmake.exe/ninja.exe*** 作為 cmake generator，Linux/Windows 開發與部署切換較不易出錯。

- **Docker 映像檔安裝**

  官方已於 docker hub 提供預建映像檔，所有 Drogon 相依套件與本身皆已安裝，可直接於 docker 環境建置 Drogon 應用。

- **Nix 套件安裝**

  Drogon 於 21.11 版釋出 Nix 套件。

> **尚未安裝 Nix 者：**請參考 NixOS 官方網站 安裝。

於專案根目錄新增 `shell.nix`，內容如下：

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

  buildInputs = with pkgs; [
    drogon
  ];

}
```

執行 `nix-shell` 進入 Drogon 開發環境。

Nix 套件可依需求調整選項：

選項	預設值
sqliteSupport	true
postgresSupport	false
redisSupport	false
mysqlSupport	false

範例：

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

- **CPM.cmake 引入**

  可用 CPM.cmake 引入 drogon 原始碼：

```
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)
```

- **專案本地引入 drogon 原始碼**

  亦可將 drogon 原始碼放於專案目錄 third_party（記得更新 submodule），於 cmake 檔加入：

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

下一步: 快速開始