第4章 ns3入门

4.1 概述

理解ns3：ns3本质上一个软件库系统。编译完成之后的ns3就是一堆库文件（libraries），每一个库完成特定的功能（这些库提供网络模拟的核心基础功能，也包括一些主流的网络模型，例如：wifi、lr-wpan、lte等网络模型）。用户利用这些库来完成网络模拟场景搭建，性能测试等工作。如果需要开发新的网络协议（或者定制现有网络协议），则可以基于现有的库来完成新网络协议开发、验证和性能测试等任务。

使用ns-3来开发网络模拟场景（或开发新的网络协议）：就是编写C++程序，在`main()`函数通过对库的调用完成网络模拟场景的配置和运行控制。此外，还经常编辑现有的ns-3库来对对相关网络协议进行定制化开发，以达到特定的目的。

ns-3的库文件列表（3.45）：

```
ns3@ns3-vm:~$ ls -l workspace/ns-allinone-3.45/ns-3.45/build/lib
total 709872
-rwxrwxr-x 1 ns3 ns3   1634160 Jul  6 12:44 libns3.45-antenna-default.so
-rwxrwxr-x 1 ns3 ns3   1570480 Jul  6 12:45 libns3.45-antenna-test-default.so
-rwxrwxr-x 1 ns3 ns3   4180088 Jul  6 13:05 libns3.45-aodv-default.so
-rwxrwxr-x 1 ns3 ns3   4850528 Jul  6 13:05 libns3.45-aodv-test-default.so
-rwxrwxr-x 1 ns3 ns3  12756712 Jul  6 13:04 libns3.45-applications-default.so
-rwxrwxr-x 1 ns3 ns3   3215288 Jul  6 13:06 libns3.45-applications-test-
default.so
-rwxrwxr-x 1 ns3 ns3   1208296 Jul  6 12:59 libns3.45-bridge-default.so
-rwxrwxr-x 1 ns3 ns3   4344680 Jul  6 13:06 libns3.45-buildings-default.so
-rwxrwxr-x 1 ns3 ns3   2665208 Jul  6 13:07 libns3.45-buildings-test-default.so
-rwxrwxr-x 1 ns3 ns3   1456096 Jul  6 13:07 libns3.45-config-store-default.so
-rwxrwxr-x 1 ns3 ns3  16704200 Jul  6 12:44 libns3.45-core-default.so
-rwxrwxr-x 1 ns3 ns3  28562864 Jul  6 13:08 libns3.45-core-test-default.so
-rwxrwxr-x 1 ns3 ns3   3187576 Jul  6 13:06 libns3.45-csma-default.so
-rwxrwxr-x 1 ns3 ns3    278568 Jul  6 13:10 libns3.45-csma-layout-default.so
-rwxrwxr-x 1 ns3 ns3   2463400 Jul  6 13:11 libns3.45-dsdv-default.so
-rwxrwxr-x 1 ns3 ns3    558104 Jul  6 13:11 libns3.45-dsdv-test-default.so
-rwxrwxr-x 1 ns3 ns3   9254160 Jul  6 13:12 libns3.45-dsr-default.so
-rwxrwxr-x 1 ns3 ns3    872464 Jul  6 13:12 libns3.45-dsr-test-default.so
-rwxrwxr-x 1 ns3 ns3   5065272 Jul  6 12:48 libns3.45-energy-default.so
-rwxrwxr-x 1 ns3 ns3    423920 Jul  6 13:12 libns3.45-energy-test-default.so
-rwxrwxr-x 1 ns3 ns3   2787360 Jul  6 13:13 libns3.45-fd-net-device-default.so
-rwxrwxr-x 1 ns3 ns3   3908520 Jul  6 13:14 libns3.45-flow-monitor-default.so
-rwxrwxr-x 1 ns3 ns3   6038520 Jul  6 13:05 libns3.45-internet-apps-default.so
-rwxrwxr-x 1 ns3 ns3   2681328 Jul  6 13:17 libns3.45-internet-apps-test-
default.so
-rwxrwxr-x 1 ns3 ns3  53973280 Jul  6 13:03 libns3.45-internet-default.so
-rwxrwxr-x 1 ns3 ns3  40484728 Jul  6 13:17 libns3.45-internet-test-default.so
-rwxrwxr-x 1 ns3 ns3   8339648 Jul  6 13:18 libns3.45-lr-wpan-default.so
-rwxrwxr-x 1 ns3 ns3   8966168 Jul  6 13:19 libns3.45-lr-wpan-test-default.so
-rwxrwxr-x 1 ns3 ns3  78636288 Jul  6 13:26 libns3.45-lte-default.so
-rwxrwxr-x 1 ns3 ns3  43997704 Jul  6 13:31 libns3.45-lte-test-default.so
-rwxrwxr-x 1 ns3 ns3  13044096 Jul  6 13:11 libns3.45-mesh-default.so
-rwxrwxr-x 1 ns3 ns3   4614328 Jul  6 13:34 libns3.45-mesh-test-default.so
-rwxrwxr-x 1 ns3 ns3   5584816 Jul  6 12:48 libns3.45-mobility-default.so
```

```
-rwxrwxr-x 1 ns3 ns3   3233920 Jul  6 13:34 libns3.45-mobility-test-default.so
-rwxrwxr-x 1 ns3 ns3   4574296 Jul  6 13:27 libns3.45-netanim-default.so
-rwxrwxr-x 1 ns3 ns3   1398504 Jul  6 13:34 libns3.45-netanim-test-default.so
-rwxrwxr-x 1 ns3 ns3  17959240 Jul  6 12:47 libns3.45-network-default.so
-rwxrwxr-x 1 ns3 ns3   6666536 Jul  6 13:36 libns3.45-network-test-default.so
-rwxrwxr-x 1 ns3 ns3   2221776 Jul  6 13:17 libns3.45-nix-vector-routing-
default.so
-rwxrwxr-x 1 ns3 ns3   1242272 Jul  6 13:36 libns3.45-nix-vector-routing-test-
default.so
-rwxrwxr-x 1 ns3 ns3   4090344 Jul  6 13:36 libns3.45-olsr-default.so
-rwxrwxr-x 1 ns3 ns3   3376760 Jul  6 13:37 libns3.45-olsr-test-default.so
-rwxrwxr-x 1 ns3 ns3   3076912 Jul  6 13:06 libns3.45-point-to-point-default.so
-rwxrwxr-x 1 ns3 ns3    869936 Jul  6 13:34 libns3.45-point-to-point-layout-
default.so
-rwxrwxr-x 1 ns3 ns3    729768 Jul  6 13:37 libns3.45-point-to-point-test-
default.so
-rwxrwxr-x 1 ns3 ns3   4874792 Jul  6 12:49 libns3.45-propagation-default.so
-rwxrwxr-x 1 ns3 ns3   3269104 Jul  6 13:37 libns3.45-propagation-test-default.so
-rwxrwxr-x 1 ns3 ns3   2901280 Jul  6 13:37 libns3.45-sixlowpan-default.so
-rwxrwxr-x 1 ns3 ns3   3517960 Jul  6 13:38 libns3.45-sixlowpan-test-default.so
-rwxrwxr-x 1 ns3 ns3  13289024 Jul  6 12:50 libns3.45-spectrum-default.so
-rwxrwxr-x 1 ns3 ns3   3844208 Jul  6 13:38 libns3.45-spectrum-test-default.so
-rwxrwxr-x 1 ns3 ns3   7718640 Jul  6 12:45 libns3.45-stats-default.so
-rwxrwxr-x 1 ns3 ns3   1430368 Jul  6 13:39 libns3.45-stats-test-default.so
-rwxrwxr-x 1 ns3 ns3   1875512 Jul  6 13:39 libns3.45-tap-bridge-default.so
-rwxrwxr-x 1 ns3 ns3   2788952 Jul  6 13:41 libns3.45-topology-read-default.so
-rwxrwxr-x 1 ns3 ns3    287520 Jul  6 13:41 libns3.45-topology-read-test-
default.so
-rwxrwxr-x 1 ns3 ns3   9236616 Jul  6 12:59 libns3.45-traffic-control-default.so
-rwxrwxr-x 1 ns3 ns3   9322952 Jul  6 13:42 libns3.45-traffic-control-test-
default.so
-rwxrwxr-x 1 ns3 ns3  10666960 Jul  6 13:27 libns3.45-uan-default.so
-rwxrwxr-x 1 ns3 ns3   1750864 Jul  6 13:43 libns3.45-uan-test-default.so
-rwxrwxr-x 1 ns3 ns3    700016 Jul  6 13:20 libns3.45-virtual-net-device-
default.so
-rwxrwxr-x 1 ns3 ns3 110755840 Jul  6 12:58 libns3.45-wifi-default.so
-rwxrwxr-x 1 ns3 ns3  87177272 Jul  6 13:49 libns3.45-wifi-test-default.so
-rwxrwxr-x 1 ns3 ns3  14433320 Jul  6 13:20 libns3.45-wimax-default.so
-rwxrwxr-x 1 ns3 ns3   1606776 Jul  6 13:49 libns3.45-wimax-test-default.so
-rwxrwxr-x 1 ns3 ns3   5491864 Jul  6 13:50 libns3.45-zigbee-default.so
-rwxrwxr-x 1 ns3 ns3   1971440 Jul  6 13:50 libns3.45-zigbee-test-default.so
-rw-rw-r-- 1 ns3 ns3     91576 Jul  6 13:59 libscratch-nested-subdir-lib.a
```

关于Linux中的`.so`文件：在 Linux操作系统系统中，`.so` 文件是 **共享对象文件（Shared Object）** 的缩写，它相当于 Windows 系统中的 `.dll`（动态链接库）文件。`.so` 文件是动态链接库的一种，用于在程序运行时提供可共享的代码和数据。

4.2 安装ns3的先决条件

安装ns3的前提，必须的软件包（Prerequisite）（以ns-3.45为例）：

先决条件	软件包/版本号
C++ compiler	clang++ or g++ (**g++ version 9 or greater**)

先决条件	软件包/版本号
Python	python3 **version >=3.8**
CMake	cmake **version >= 3.13**
Build system	make, ninja, xcodebuild(XCode)
Git	any recent version (to access ns-3 from GitLab.com)
tar	any recent version (to unpack an ns-3 release)
bunzip2	any recent version (to uncompress an ns-3 release)

```
$ sudo apt install build-essential cmake git vim
$ sudo apt install ninja-build
```

检查相关软件的版本：

```
g++ -v
python3 -V
cmake --version
```

注意：以上仅满足ns-3基本功能的安装，ns-3的一些特定功能需要其他第三方软件或库的支持，需要另外安装，详见：https://www.nsnam.org/docs/release/3.45/installation/html/index.html）：

4.2.1 下载ns3源码包

ns-3.45提供了两个源码压缩包（https://www.nsnam.org/releases/）：

1. `ns-3.45.tar.bz2`：只包括ns-3源码
2. `ns-allinone-3.45.tar.bz2`：除了ns-3源码之外，还包括了一些与ns-3.45兼容的扩展模块（ns-3 App Store）
    1. allinone压缩包包括了一些contributions网络模型，要编译这些模块，需要安装：
    2. `sudo apt install libboost-system-dev libboost-thread-dev`

4.3 使用git下载ns3（可选）

```
$ cd
$ mkdir workspace
$ cd workspace
$ git clone https://gitlab.com/nsnam/ns-3-dev.git
$ cd ns-3-dev
```

4.4 编译ns3

ns-3 从 版本 3.36（2021 年 6 月发布）开始正式引入 CMake 作为默认构建工具，同时逐步弃用原有的 Waf 构建工具。

ns-3.45已经完全使用cmake构建工具。为了方便起见，ns-3提供了一个CMake包装器`ns3`（在源码目录下），绝大多数的最终用户不需要直接使用`cmake`命令。

4.4.1 使用`ns3`(CMake wrapper)编译(重点掌握)

编译带debug信息的ns3（初学、开发阶段）：

```
$ ./ns3 clean    # 删除`build/`目录中先前编译的库和对象文件
$ ./ns3 configure --build-profile=debug --enable-examples --enable-tests
$ ./ns3 build    # 编译ns-3
```

编译优化的ns3（发布阶段）：

```
$ ./ns3 clean    # 删除`build/`目录中先前配置和编译的生成配置文集、库和对象文件，可选
$ ./ns3 configure --build-profile=optimized --enable-examples --enable-tests
$ ./ns3 build    # 编译ns-3
```

使用如下命令来查看当前的配置是`debug`还是`optimized`：

```
$ ./ns3 show profile
Build profile: debug
```

一些有用的配置选项：

```
$ ./ns3 configure -h
usage: ns3 configure [-h] [-d {debug,default,release,optimized,minsizerel}]
                     [-G G] [--cxx-standard CXX_STANDARD] [--enable-asserts]
                     [--disable-asserts] [--enable-des-metrics]
                     [--disable-des-metrics] [--enable-build-version]
                     [--disable-build-version] [--enable-clang-tidy]
                     [--disable-clang-tidy] [--enable-dpdk] [--disable-dpdk]
                     [--enable-eigen] [--disable-eigen] [--enable-examples]
                     [--disable-examples] [--enable-gcov] [--disable-gcov]
                     [--enable-gsl] [--disable-gsl] [--enable-gtk]
                     [--disable-gtk] [--enable-logs] [--disable-logs]
                     [--enable-monolib] [--disable-monolib] [--enable-mpi]
                     [--disable-mpi] [--enable-ninja-tracing]
                     [--disable-ninja-tracing] [--enable-precompiled-headers]
                     [--disable-precompiled-headers]
                     [--enable-python-bindings] [--disable-python-bindings]
                     [--enable-tests] [--disable-tests] [--enable-sanitizers]
                     [--disable-sanitizers] [--enable-static]
                     [--disable-static] [--enable-sudo] [--disable-sudo]
                     [--enable-verbose] [--disable-verbose]
                     [--enable-warnings] [--disable-warnings]
                     [--enable-werror] [--disable-werror]
                     [--enable-modules ENABLE_MODULES]
                     [--disable-modules DISABLE_MODULES]
                     [--filter-module-examples-and-tests
FILTER_MODULE_EXAMPLES_AND_TESTS]
                     [--lcov-report] [--lcov-zerocounters]
                     [--out OUTPUT_DIRECTORY] [--with-brite WITH_BRITE]
                     [--with-click WITH_CLICK] [--with-openflow WITH_OPENFLOW]
                     [--force-refresh] [--prefix PREFIX] [--trace-performance]
                     [--dry-run] [--quiet] [-v]
```

```
positional arguments:
  configure

options:
  -h, --help            show this help message and exit
  -d {debug,default,release,optimized,minsizerel}, --build-profile
{debug,default,release,optimized,minsizerel}
                        Build profile
  -G G                  CMake generator (e.g.
                        https://cmake.org/cmake/help/latest/manual/cmake-
                        generators.7.html)
  --cxx-standard CXX_STANDARD
                        Compile NS-3 with the given C++ standard
  --enable-asserts      Enable the asserts regardless of the compile mode
  --disable-asserts     Disable the asserts regardless of the compile mode
  --enable-des-metrics  Enable Logging all events in a json file with the name
                        of the executable (which must call
                        CommandLine::Parse(argc, argv))
  --disable-des-metrics
                        Disable Logging all events in a json file with the
                        name of the executable (which must call
                        CommandLine::Parse(argc, argv))
  --enable-build-version
                        Enable embedding git changes as a build version during
                        build
  --disable-build-version
                        Disable embedding git changes as a build version
                        during build
  --enable-clang-tidy   Enable clang-tidy static analysis
  --disable-clang-tidy  Disable clang-tidy static analysis
  --enable-dpdk         Enable the fd-net-device DPDK features
  --disable-dpdk        Disable the fd-net-device DPDK features
  --enable-eigen        Enable Eigen3 library support
  --disable-eigen       Disable Eigen3 library support
  --enable-examples     Enable the ns-3 examples
  --disable-examples    Disable the ns-3 examples
  --enable-gcov         Enable code coverage analysis
  --disable-gcov        Disable code coverage analysis
  --enable-gsl          Enable GNU Scientific Library (GSL) features
  --disable-gsl         Disable GNU Scientific Library (GSL) features
  --enable-gtk          Enable GTK support in ConfigStore
  --disable-gtk         Disable GTK support in ConfigStore
  --enable-logs         Enable the logs regardless of the compile mode
  --disable-logs        Disable the logs regardless of the compile mode
  --enable-monolib      Enable a single shared library with all ns-3 modules
  --disable-monolib     Disable a single shared library with all ns-3 modules
  --enable-mpi          Enable the MPI support for distributed simulation
  --disable-mpi         Disable the MPI support for distributed simulation
  --enable-ninja-tracing
                        Enable the conversion of the Ninja generator log file
                        into about://tracing format
  --disable-ninja-tracing
                        Disable the conversion of the Ninja generator log file
                        into about://tracing format
  --enable-precompiled-headers
                        Enable precompiled headers
```

```
  --disable-precompiled-headers
                        Disable precompiled headers
  --enable-python-bindings
                        Enable python bindings
  --disable-python-bindings
                        Disable python bindings
  --enable-tests        Enable the ns-3 tests
  --disable-tests       Disable the ns-3 tests
  --enable-sanitizers   Enable address, memory leaks and undefined behavior
                        sanitizers
  --disable-sanitizers  Disable address, memory leaks and undefined behavior
                        sanitizers
  --enable-static       Build a single static library with all ns-3
  --disable-static      Restore the shared libraries
  --enable-sudo         Enable use of sudo to setup suid bits on ns3
                        executables.
  --disable-sudo        Disable use of sudo to setup suid bits on ns3
                        executables.
  --enable-verbose      Enable printing of additional build system messages
  --disable-verbose     Disable printing of additional build system messages
  --enable-warnings     Enable compiler warnings
  --disable-warnings    Disable compiler warnings
  --enable-werror       Treat compiler warnings as errors
  --disable-werror      Treat compiler warnings as warnings
  --enable-modules ENABLE_MODULES
                        List of modules to build (e.g.
                        "core;network;internet")
  --disable-modules DISABLE_MODULES
                        List of modules not to build (e.g. "lte;wimax")
  --filter-module-examples-and-tests FILTER_MODULE_EXAMPLES_AND_TESTS
                        List of modules that should have their examples and
                        tests built (e.g. "lte;wifi")
  --lcov-report         Generate a code coverage report (use this option after
                        configuring with --enable-gcov and running a program)
  --lcov-zerocounters   Zero the lcov counters (use this option before
                        rerunning a program when generating repeated lcov
                        reports)
  --out OUTPUT_DIRECTORY, --output-directory OUTPUT_DIRECTORY
                        Directory to store build artifacts
  --with-brite WITH_BRITE
                        Use BRITE integration support, given by the indicated
                        path, to allow the use of the BRITE topology generator
  --with-click WITH_CLICK
                        Path to Click source or installation prefix for NS-3
                        Click Integration support
  --with-openflow WITH_OPENFLOW
                        Path to OFSID source for NS-3 OpenFlow Integration
                        support
  --force-refresh       Force refresh the CMake cache by deleting the cache
                        and reconfiguring the project
  --prefix PREFIX       Target output directory to install
  --trace-performance   Generate a performance trace log for the CMake
                        configuration
  --dry-run             Do not execute the commands.
  --quiet               Don't print task lines, i.e. messages saying which
                        tasks are being executed.
```

```
  -v, --verbose         Print which commands were executed
```

下面介绍几个可能会用到的配置选项：

(1) `--build-profile` 或 `-d`

可以有三个选项：debug、default、release、optimized。

Build profiles

Feature	Build Profile			
	debug	default	release	optimized
Enabled Features	NS3_BUILD_PROFILE_DEBUG NS_LOG... NS_ASSERT...	NS3_BUILD_PROFILE_DEBUG NS_LOG... NS_ASSERT...	NS3_BUILD_PROFILE_RELEASE	NS3_BUILD_PROFILE_OPTIMIZED
Code Wrapper Macro	NS_BUILD_DEBUG(code)	NS_BUILD_DEBUG(code)	NS_BUILD_RELEASE(code)	NS_BUILD_OPTIMIZED(code)
Compiler Flags	-Og -g	-Os -g	-O3	-O3 -march=native -mtune=native

注意：日志和断言功能默认只在debug和default时可用，release和optimized则需要显式的通过配置选项（`--enable-logs`，`--enable-asserts`）来激活这两个功能。

(2) `--out`

默认情况下，ns3编译后将保存在build目录中。如果要指定其他目录，使用如下的配置（使用`--out`配置选项）：

```
$ ./ns3 configure --out=my-build-dir
```

`--out`和`--build-profile`结合可以编译两个版本（`debug`，`optimized`）的ns3，在开发时使用debug，发布时使用optimized：

```
$ ./ns3 configure --build-profile=debug --out=build/debug
$ ./ns3 build
...
$ ./ns3 configure --build-profile=optimized --out=build/optimized
$ ./ns3 build
```

When you switch, ns3 will only compile what it has to, instead of recompiling everything.

(3) `--enable-sudo`

需要使用ns-3的仿真特性的用户，需要获得root权限。通过`--eanble-sodu`来实现此目的。

```
$ ./ns3 configure --enable-sudo --enable-examples --enable-tests
```

(4) `--dry-run`

该选项不属于`configure`，但是比较有用。仅查看命令将要干什么，而不真正执行该命令。

(5) 使用其他编译和指定编译器标志（可选）：

上面的示例中默认使用GCC的C++编译器，要使用其他编译器（例如：Clang C++编译器）：

```
$ CXX="clang++" ./ns3 configure
$ ./ns3 build

分布式编译：
$ CXX="distcc g++" ./ns3 configure
$ ./ns3 build
```

如果在编译时要添加编译器标志，类似上面的例子，在配置ns-3时使用`CXXFLAGS_EXTRA`环境变量。

通过环境变量简化设置（可选）：

通过定义环境变量可以在输入配置命令的时候避免出错：

```
$ export NS3CONFIG="--enable-examples --enable-tests"
$ export NS3DEBUG="--build-profile=debug --out=build/debug"
$ export NS3OPT=="--build-profile=optimized --out=build/optimized"
$ ./ns3 configure $NS3CONFIG $NS3DEBUG
$ ./ns3 build
...
$ ./ns3 configure $NS3CONFIG $NS3OPT
$ ./ns3 build
```

编译后的安装操作（可选）

编译完成后可以使用`$ ./ns3 install`或`$ sudo ./ns3 install`将构建的文件复制到系统目录中（**ns3编译后的库和可执行程序默认被放到build目录下，ns3知道这些库和可执行程序的位置，因此不需要安装到其他位置**），如果配置时指定了安装目录（`./ns3 configure --prefix=/opt/local`）则复制到指定的安装目录。

通常，我们可以不执行安装命令，每次直接进入ns-3目录下完成相关工作。

清理操作（可选）

清理指的是删除构建过程生成或编辑的文件，有如下几种方式：

Scope	Command	Description
clean	./ns3 clean	Remove artifacts generated by the CMake configuration and the build
distclean	./ns3 distclean	Remove artifacts from the configuration, build, documentation, test and Python
ccache	ccache -C	Remove all compiled artifacts from the ccache

- 如果重点是重新配置当前编译ns-3的方式，则可以使用`clean`。
- 如果重点是将ns-3目录恢复到原始状态，则可以使用`distclean`。
- **关于ccache：**
  - ccache是单独的命令，不从属于ns-3。**ccache**（**compiler cache**，编译器缓存）是一个开源的 **编译加速工具**，主要用于 **C/C++** 编译过程，通过 **缓存编译结果** 来 **减少重复编译的时间**，从而显著加快 **增量编译（如修改少量代码后重新编译）** 的速度。
  - 如果系统中安装了ccache，ns-3在编译时会自动使用ccache来加快编译。ccache缓存的文件位于ns-3目录之外（通常位于`~/.cache/ccache`的隐藏目录中），并在多个项目之间共享。

- 清理缓存将导致当前工作目录之外的其他构建目录上的缓存丢失。定期清理此缓存可能有助于回收磁盘空间。
- 清理ccache与清理ns-3目录中的任何文件完全不同。

为了确保清理的安全性，可以使用如下的命令事先看看要执行啥操作，然后再真正执行的清理：

```
./ns3 clean --dry-run
```

ns3便捷设置：one ns3

经常需要`$ ../../../ns3`这样书写命令，麻烦！

```
$ export NS3DIR="$PWD"
$ function ns3f { cd $NS3DIR && ./waf $* ; }
$ cd scratch
$ ns3f build
```

与安装有关的命令小结：

> 配置：
> ./ns3 configure --prefix=/opt/local
> ./ns configure --build-profile=debug --enable-examples --enable-tests
> 编译：./ns3 build
> 安装与反安装：
> ./ns3 install
> ./ns3 clean --dry-run # 删除由CMake生成的配置文件和build目录下的内容
> ./ns3 distclean #将ns-3目录恢复到原始状态
> 查看当前有效的配置文件类型：
> ./ns3 show profile #显示debug，default，release，optimized
> ./ns3 show config #显示模块配置：哪些激活、哪些禁用

4.4.2 使用CMake编译（可选，不推荐）

不推荐直接使用cmake来完成编译工作。

ns3本质上是CMake的wrapper，它会自动调用`cmake`命令完成所有的构建工作，使用`--dry-run`可以看到幕后的真正命令。

```
$ ./ns3 configure --enable-tests --enable-examples -d optimized
$ ./ns3 configure --enable-tests --enable-examples --build-profile=optimized
上面两个命令实际执行的是下面的命令：
$ cd /ns-3-dev/cmake-cache/
$ cmake -DCMAKE_BUILD_TYPE=release -DNS3_NATIVE_OPTIMIZATIONS=ON -DNS3_ASSERT=OFF
-
,!DNS3_LOG=OFF -DNS3_TESTS=ON -DNS3_EXAMPLES=ON ..

再比如：
ns3@ns3-vm:~/workspace/ns-allinone-3.45/ns-3.45$ ./ns3 build test-runner
上面的命令实际执行的是下面的命令：
Finished executing the following commands:
/usr/bin/cmake --build /home/ns3/workspace/ns-allinone-3.45/ns-3.45/cmake-cache -
j 1 --target test-runner
```

使用ns3运行网络模拟程序时，幕后也要调用cmake命令：

```
ns3@ns3-vm:~/workspace/ns-allinone-3.45/ns-3.45$ ./ns3 run first
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9

ns3@ns3-vm:~/workspace/ns-allinone-3.45/ns-3.45$ ./ns3 run first --dry-run
# --dry-out 表示并不实际执行命令
The following commands would be executed:
/usr/bin/cmake --build /home/ns3/workspace/ns-allinone-3.45/ns-3.45/cmake-cache -
j 1 --target first
cd .; export PATH=$PATH:/home/ns3/workspace/ns-allinone-3.45/ns-3.45/build/lib
PYTHONPATH=/home/ns3/workspace/ns-allinone-3.45/ns-3.45/build/bindings/python
LD_LIBRARY_PATH=/home/ns3/workspace/ns-allinone-3.45/ns-3.45/build/lib ;
/home/ns3/workspace/ns-allinone-3.45/ns-3.45/build/examples/tutorial/ns3.45-
first-default
```

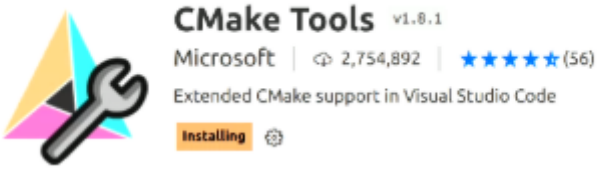
4.4.4 使用IDE编译ns3

由于使用了CMake，ns-3与IDE的整合更加容易。

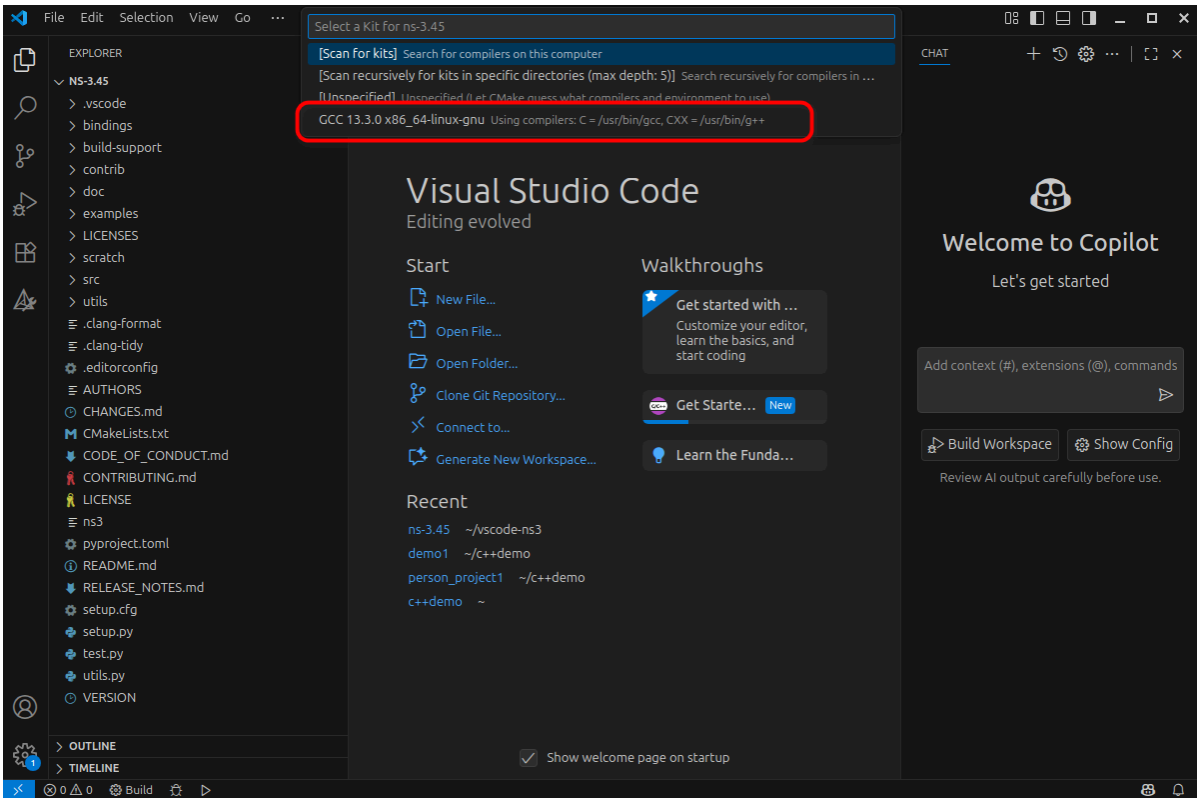
下面以Microsoft Visual Studio Code为例介绍如何使用IDE编译ns3.

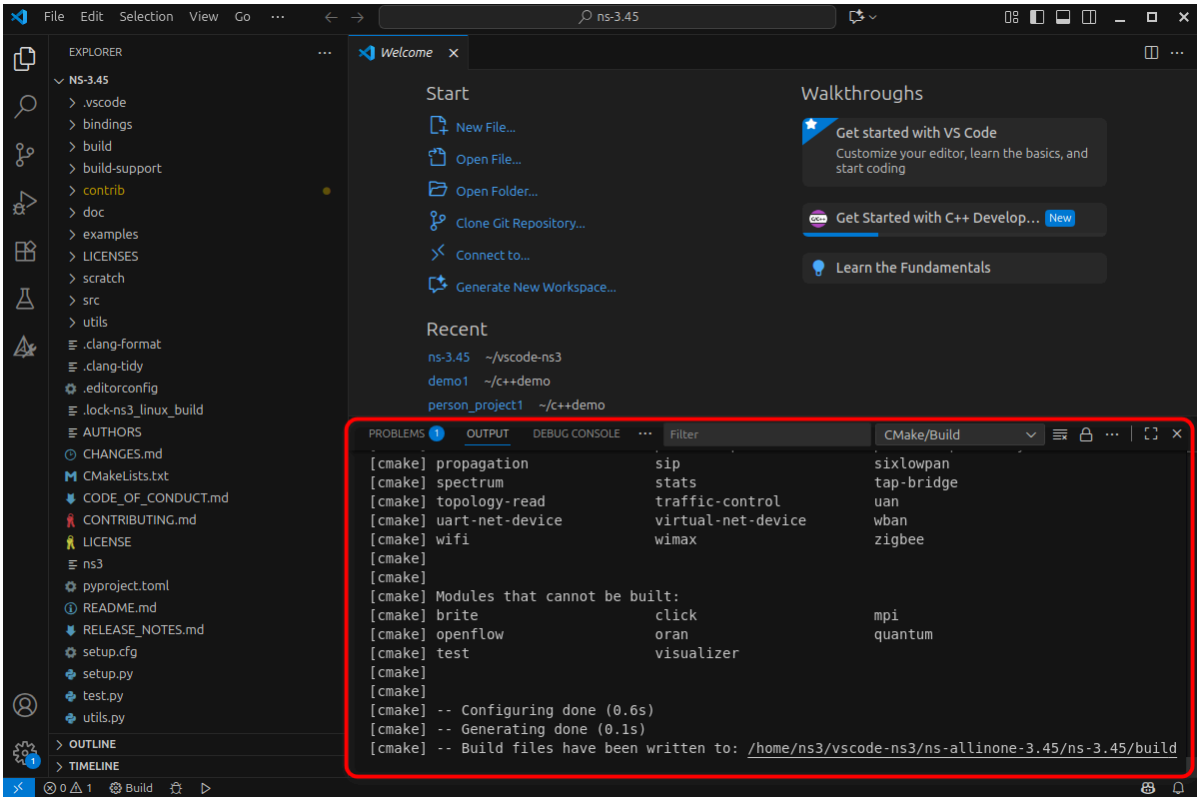
Microsoft Visual Studio Code：

- 安装VS Code:
- 安装VS Code 插件：C++、CMake Tools

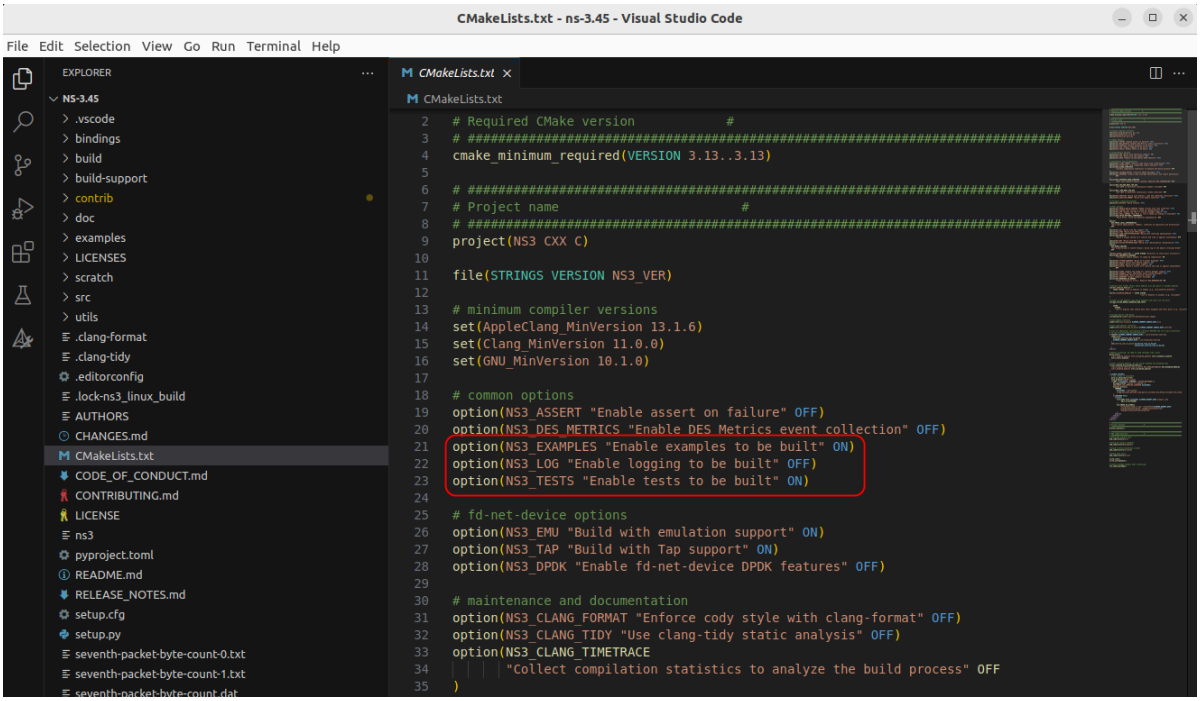


- 使用VS Code打开ns-3的源码目录。**注意**：因为安装了`CMake Tools`插件，VS Code会自动运行预配置，如下图所示：





- 使用VS Code打开ns-3的源码目录下的`CMakeLists.txt`，找到对应的`option`配置开关，打开或者关闭相关功能模块，如下图所示：

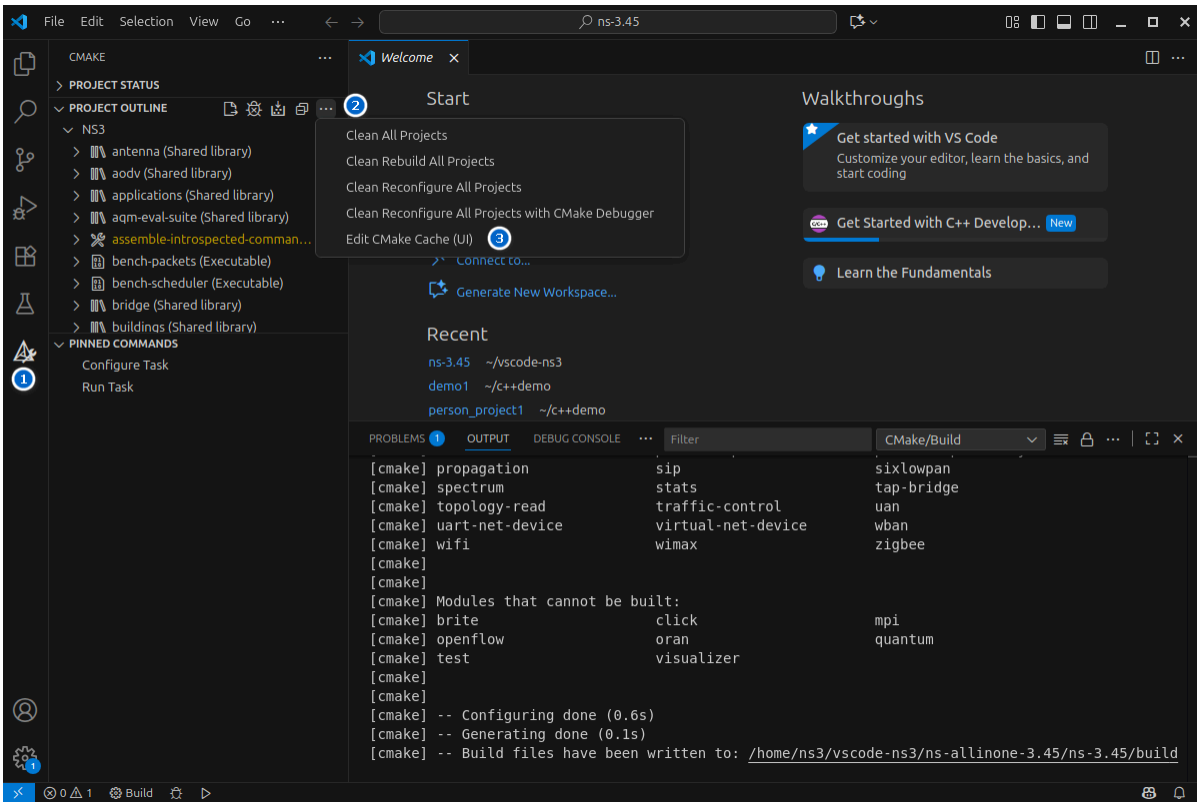


注意：当保存对`CMakeLists.txt`修改后，自动运行配置命令。由于使用VS Code打开ns-3源码目录时，已经自动运行了配置，相关的配置变量已经保存在了`build\CMakeCache.txt`中，因此，对CMakeLists.txt的修改不会重新配置更新`CMakeCache.txt`文件。要使上面的配置生效，需要删除缓存文件，然后再次保存，即可重新生成`CMakeCache.txt`。

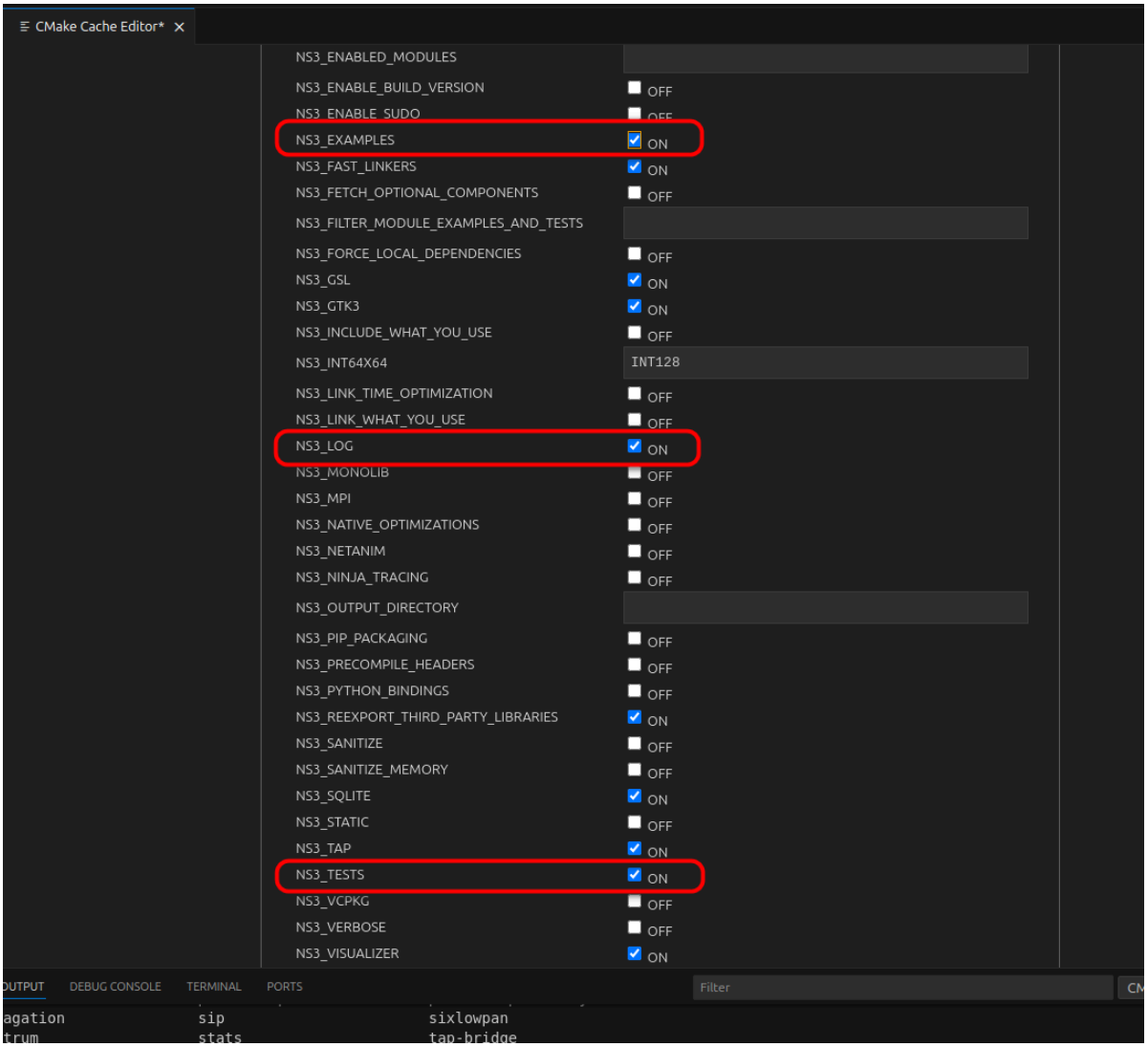
这是因为，通过`option`设置变量的值是`initial_value`，它只在第一次配置或者缓存中还没有该变量时生效。即仅当该变量 **尚未在 CMake 缓存中设置时** 才会使用！**解决方法：**最简单的方法，删除构建目录（如`build/`），然后重新创建并配置：

```
$ ./ns3 clean --dry-run
The following commands would be executed:
rm -R build
rm -R build
rm -R build
rm -R .lock-ns3_linux_build
```

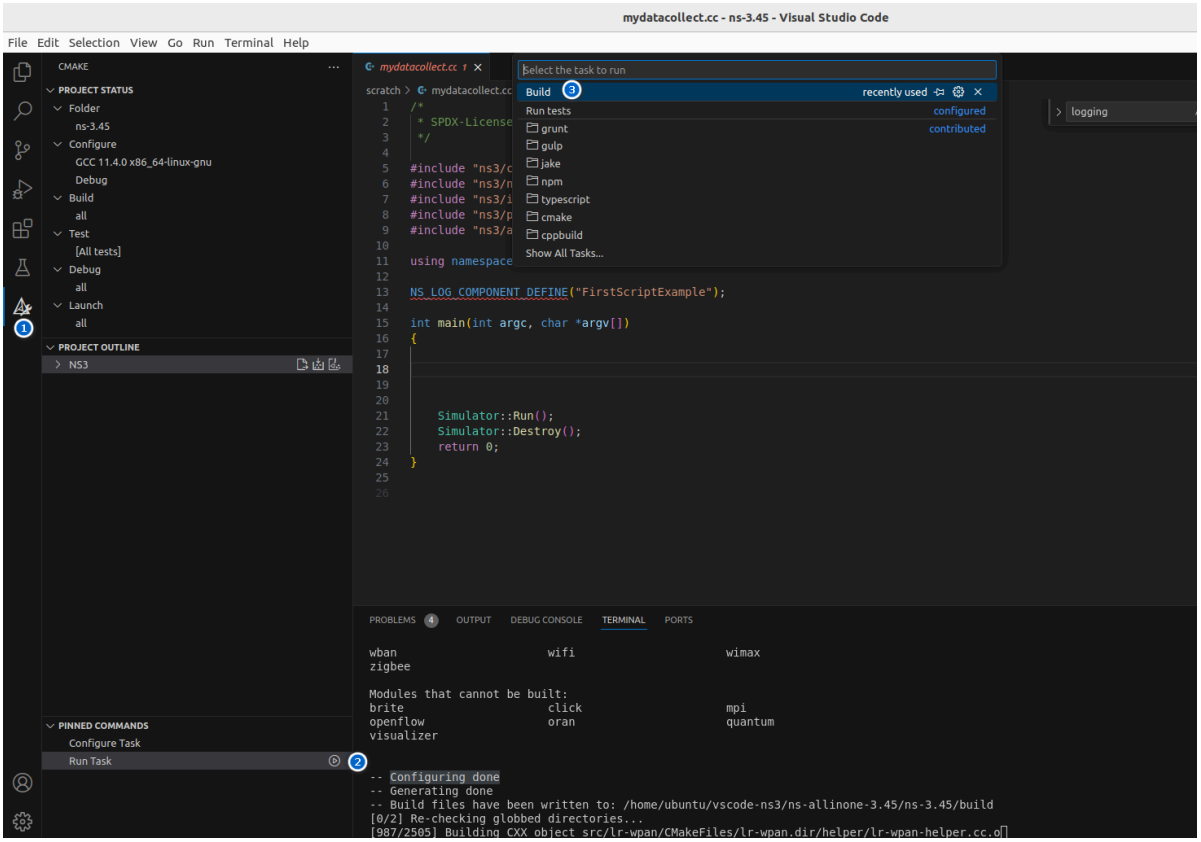
或者使用下图的方式打开`CMakeCache.txt`：



打开`NS3_EXAMPLES`，`NS3_LOG`，`NS3_TESTS`开关并保存，如下图所示：

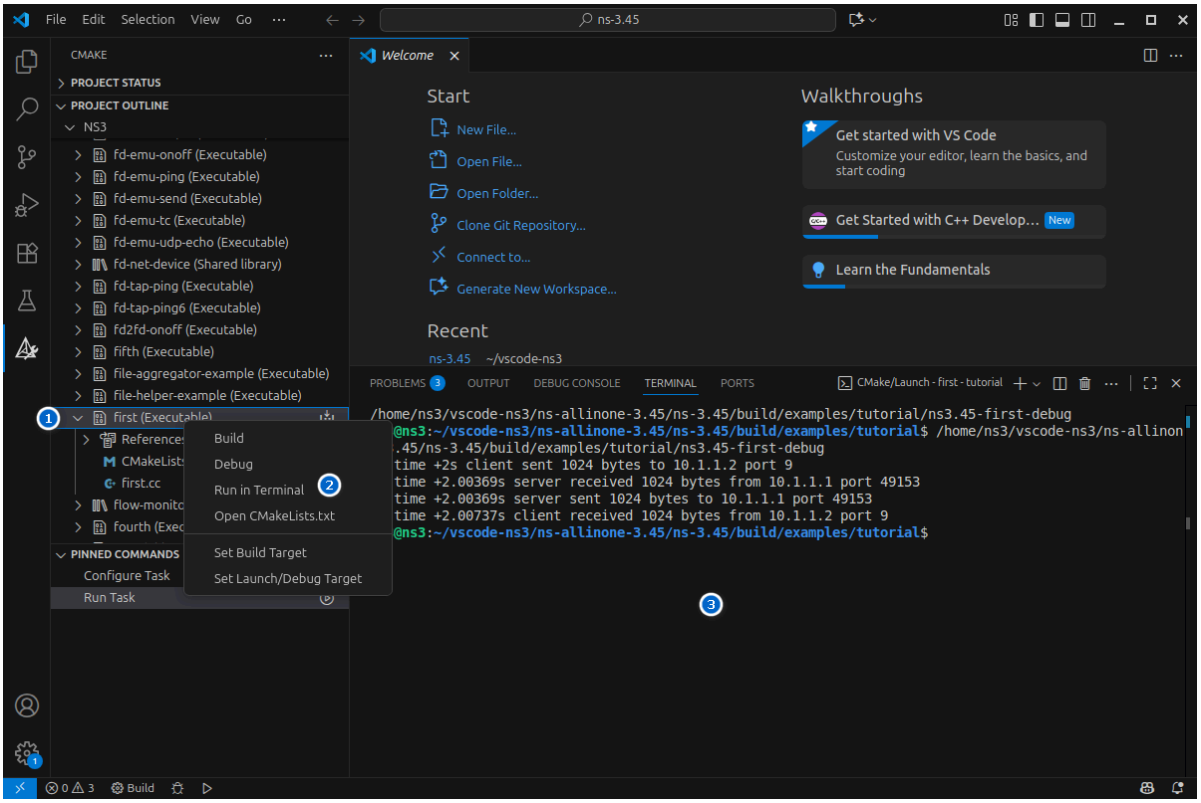


- 编译ns-3：在终端中运行`./ns3 build`或者使用`CMake Tools`：

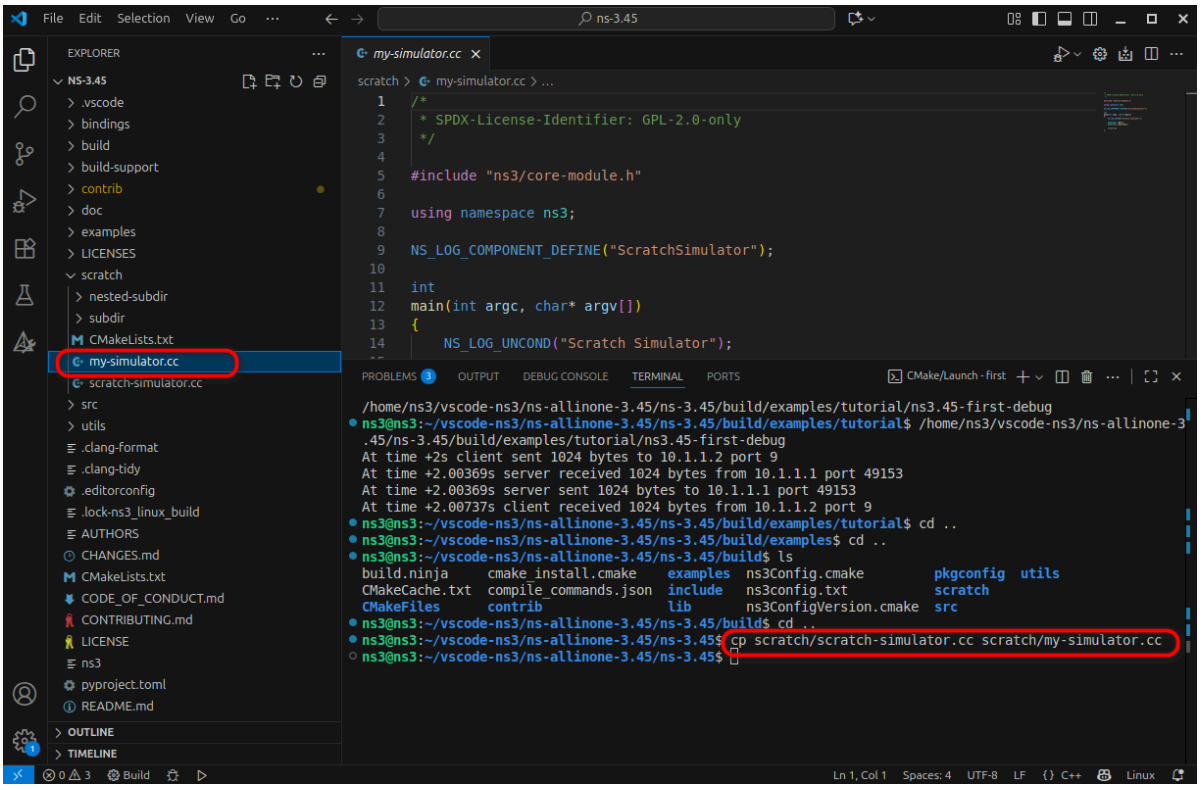


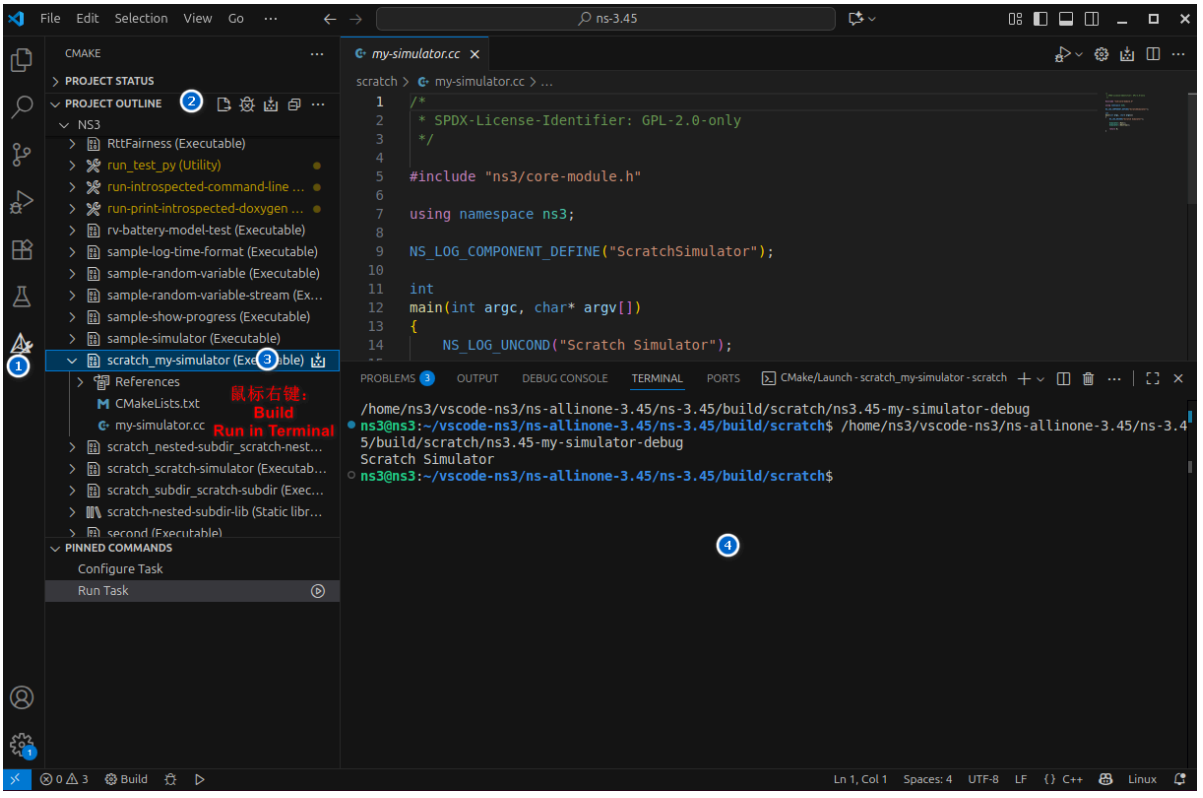
构建、运行网络模拟程序

在VS Code可以使用`CMake Tools`插件提供的图形化界面来构建和运行网络模拟程序，如下图所示：



添加自己的网络模拟程序：





4.5 测试已构建ns-3

验证安装的ns-3是否正确的完成了构建。

```
$ cd ns-allinone-3.45/ns-3.45/
$./test.py               # 在运行测试之前，先检查是否需要构建（编译）ns-3
$./test.py --no-build    # 跳过构建，仅运行测试
```

4.6 运行网络模拟脚本

我们通常在ns3的控制下运行网络模拟脚本，这种方式通过构建系统来确保在运行脚本时，共享库路径的配置是正确的，共享库是可用的。

运行ns3的`Hello Simulator`程序：

```
./ns3 run hello-simulator            #hello-simator源代码已被正确编译
Hello Simulator

./ns3 run hello-simulator            #修改hello-simulator后运行
[  0%] Building CXX object examples/tutorial/CMakeFiles/hello-
simulator.dir/hello-simulator.cc.o
[  0%] Linking CXX executable ns3.45-hello-simulator-debug
Hello Simulator!

./ns3 run hello-simulator --no-build  #运行现有的二级制程序，不检查源代码文件时间戳
Hello Simulator
```

注意：ns3被编译为`optimized`时看不到任何输出，因为在此模式下`logging component`在控制台的输出会被忽略。此时需要将`optimized`改为`debug`，使用如下命令：

```
./ns3 configure --build-profile=debug --enable-examples --enable-tests
./ns3  #the same to:  ./ns3 build
```

4.6.1 命令行参数

向模拟脚本中传递命令行参数：

> ```
> $ ./ns3 run <ns3-program> --command-template="%s <args>"
> ```

其中：

- `ns3-program`：要运行的网络模拟程序。
- `--command-template`：用于构造要运行的程序及其命令行参数。
  - `%s`：被ns3程序替换成`ns3-program`
  - `args`：要传递给`ns3-program`的参数列表

执行过程：

- ns3首先检查`ns3-program`是否被正确编译，如果未编译，则先完成编译。除非使用了`--no-build`，才会运行老版本的程序。
- 设置共享库路径。
- 运行网络模拟程序。

下面的例子单独运行测试套件(之前使用的`test.py`就是调用的`test-runner`)：

```
./ns3 run test-runner --command-template="%s --suite=mytest --verbose"   #报错，因为mytest并不存在
```

上面的命令行参数比较繁琐，下面的用法更加简洁：

```
./ns3 run '<ns3-program> --arg1=value1 --arg2=value2 ...‘
./ns3 run "<ns3-program> --arg1=value1 --arg2=value2 ...“
```

4.6.2 调试

当需要在其他工具控制下来运行ns-3程序（例如：`gdb`调试器，`valgrind`内存检查器），则需要使用下面的命令行形式：

```
$ ./ns3 run <ns3-program> --command-template="%s <args>"
```

使用gdb调试程序：

```
./ns3 run hello-simulator --command-template="gdb %s --args <args>"
./ns3 run test-runner --command-template="gdb %s --args --suite=mytest --verbose"
```

其中，`--args`表示后面的`args`是要传递给被调试程序`hello-simulator`的参数。

4.6.3 指定工作目录

通常，我们在ns-3目录下运行模拟程序，此时模拟程序输出的文件都保存在ns-3目录下，通过下面的方式可以将输出文件保存到指定位置：

```
# 指定工作目录，输出文件将存放至--cwd指定的目录下。
./ns3 run program-name --cwd=...
```

4.6.4 不编译直接运行现有程序

跳过编译，直接运行程序。

```
$ ./ns run first --no-build
$ ./ns3 run '<ns3-program> --arg1=value1 --arg2=value2 ...' --no-build
```

4.6.5 构建版本(略)

在ns-3.32版本中，引入了一个新的ns3配置选项`--enable-build-version`，它会在构建期间检查本地ns3 git存储库，并将版本元数据添加到核心模块中。

```
$ ./ns3 show version
Build version feature disabled. Reconfigure ns-3 with ./ns3 configure --enable-build-version
```

上面的输出表明，构建时没有使用`--enable-build-version`，无法显示版本信息。

4.6.6 源代码版本(略)

在ns-3库中存储构建版本信息的另一种方法是跟踪用于构建代码的源代码版本。在使用Git时，可以将以下方法添加到Bash shell脚本中，以创建包含Git修订信息的version.txt文件，如果存储库不完整，则将该版本的任何更改附加到该文件的补丁中。然后可以将生成的文本文件与任何相应的ns-3模拟结果一起保存。

```
echo `git describe` > version.txt
gitDiff=`git diff`
if [[ $gitDiff ]]
then
    echo "$gitDiff" >> version.txt
fi
```