University of Minho

MSc. Engineering of Computer Networks and Telematic Services

ndnSIM 2.3 Installation Manual

2017

Ertugrul Dogruluk

Contents

1 NS-3 Introduction

ns-3 is a discrete-event network simulator, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.
The goal of the ns-3 project is to develop a preferred, open simulation environment for networking research: it should be aligned with the simulation needs of modern networking research and should encourage community contribution, peer review, and validation of the software.

1.1 Simulating the Models

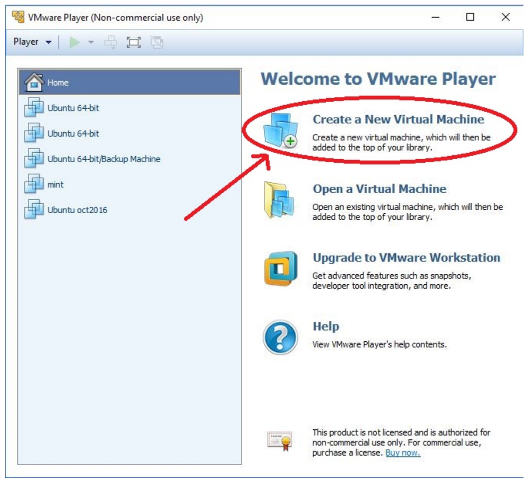
The ns-3 project is committed to building a solid simulation core that is well documented, easy to use and debug, and that caters to the needs of the entire simulation workflow, from simulation configuration to trace collection and analysis.
Furthermore, the ns-3 software infrastructure encourages the development of simulation models which are sufficiently realistic to allow ns-3 to be used as a realtime network emulator, interconnected with the real world and which allows many existing real-world protocol implementations to be reused within ns-3.
The ns-3 simulation core supports research on both IP and non-IP based networks. However, the large majority of its users focuses on wireless/IP simulations which involve models for Wi-Fi, WiMAX, or LTE for layers 1 and 2 and a variety of static or dynamic routing protocols such as OLSR and AODV for IP-based applications.
ns-3 also supports a real-time scheduler that facilitates a number of "simulation-in-the-loop" use cases for interacting with real systems. For instance, users can emit and receive ns-3-generated packets on real network devices, and ns-3 can serve as an interconnection framework to add link effects between virtual machines. Another emphasis of the simulator is on the reuse of real application and kernel code. Frameworks for running unmodified applications or the entire Linux kernel networking stack within ns-3 are presently being tested and evaluated.

2 Ubuntu Installation

You have two different options to install ubuntu on you machine; using virtual drive and dual-boot. The dual-boot option is preferable to have good performance on your machines but before doing that, backup all your files in case.

2.1 VMware-Virtual Machine

You may prefer to use virtual machine which you can run it on your current operating system(such as Win 10, Macintosh). To install and run Linux or any others, you need to have VMware software. The VMware is a free software and you may get it from VMware Player download page. Once you get it and installed to your system, you are able to do installation of Linux machine.


VMware Player (Non-commercial use only)
Player
Home
Ubuntu 64-bit
Ubuntu 64-bit
Ubuntu 64-bit/Backup Machine
mint
Ubuntu oct2016
Welcome to VMware Player
Create a New Virtual Machine
Open a Virtual Machine
Upgrade to VMware Workstation
Help
Buy now.

Create a new virtual machine, which will then be added to the top of your library.

Open an existing virtual machine, which will then be added to the top of your library.

Get advanced features such as snapshots, developer tool integration, and more.

View VMware Player's help contents.

This product is not licensed and is authorized for non-commercial use only. For commercial use, purchase a license. Buy now.

Figure 1: Create a new virtual Machine

2.2 Dual-Boot Windows 10 and Ubuntu

If you prefer to install/run Linux system physically, you need to do "create partition" from your current OS disk. To download and install Ubuntu Linux alongside windows 8&10, you need to follow the procedures from this web page Install Ubuntu Alongside Windows 10. Important: Before doing this steps, please **backup** your files.

3 Eclipse Installation

To edit and modify any scenario you need Eclipse editor on your ubuntu. The `java` is required to install Eclipse.

First of all you need to download/install a java by following;

```
sudo apt-get install python-software-properties
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
java-version
sudo apt install default-jre
sudo apt-get update
```

Then, you need to download/run the eclipse installer from this eclipse installer tool.

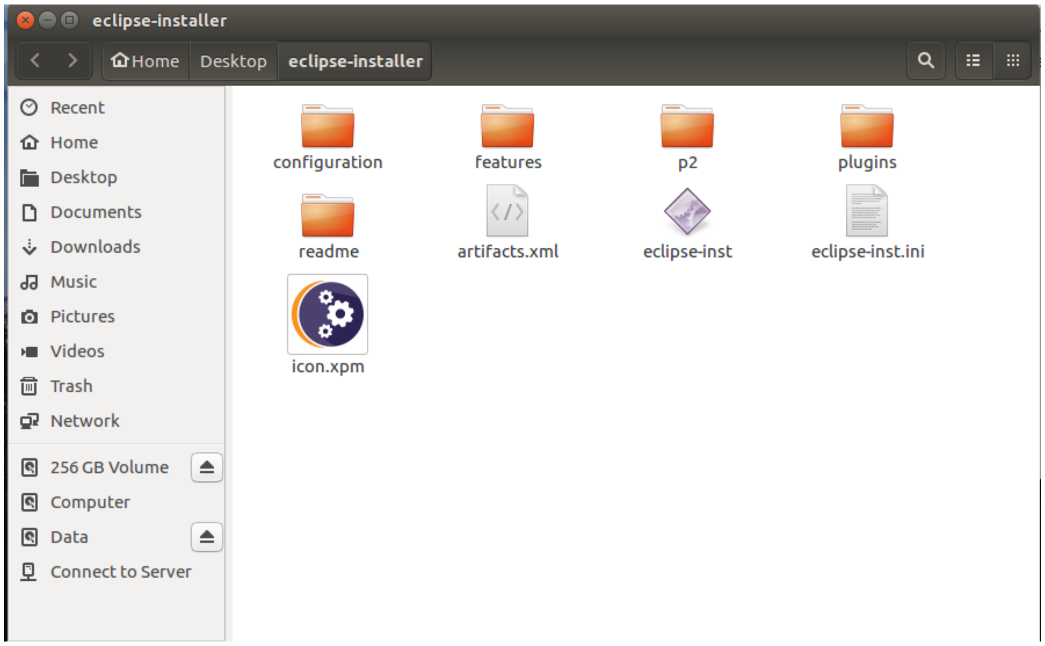
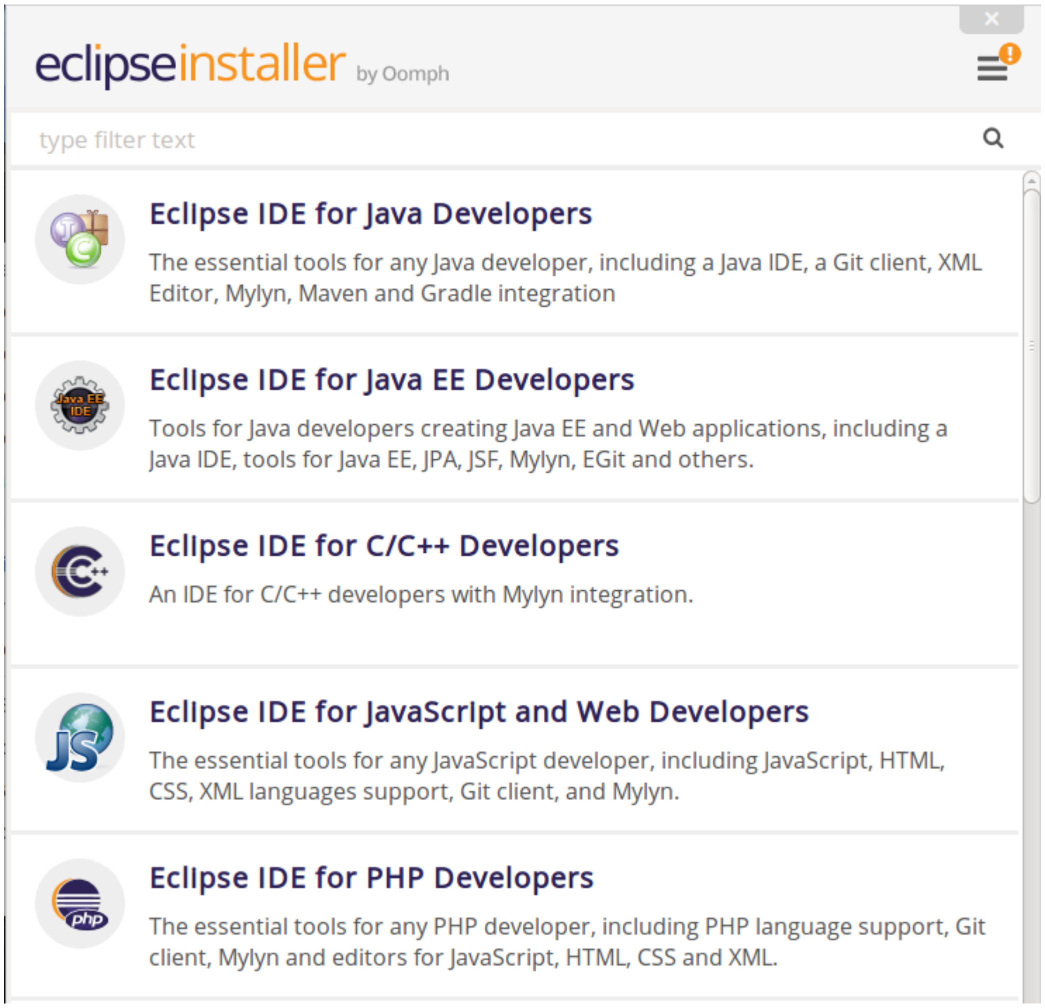


Figure 2: eclipse installation

On installer page, you need to select `Eclipse IDE for C/C++ Developers` to edit ndnsim or ns-3 files.



Eclipse IDE for Java Developers

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

Eclipse IDE for Java EE Developers

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.

Eclipse IDE for C/C++ Developers

An IDE for C/C++ developers with Mylyn integration.

Eclipse IDE for JavaScript and Web Developers

The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, and Mylyn.

Eclipse IDE for PHP Developers

The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Figure 3: eclipse installation

4 Compiling ndnSIM

To do compiling the latest version of ndnSIM, please follow the steps from this ndnSIM 2.3 Getting Started on your **Ubuntu 16.04, Mac and Fedora**. Remind that; the less problematic system is Ubuntu, so prefer to do compiling on Ubuntu 16.04. The location of ndnSIM folder can be anywhere you want. I'm personally, create my ndnSIM or ns-3 directories on my Desktop. The full compiling could take between 1-2.5 hours (depends).

5 Pulling ndnSIM directory into Eclipse

Firstly, click the new on left corner. Then the eclipse will ask you what type of project you like to create. You need to select `C++ project` under the C/C++ section.

Once you complete this process, the Eclipse will start to Indexing the project that will appear on right bottom of window. This will take some time.

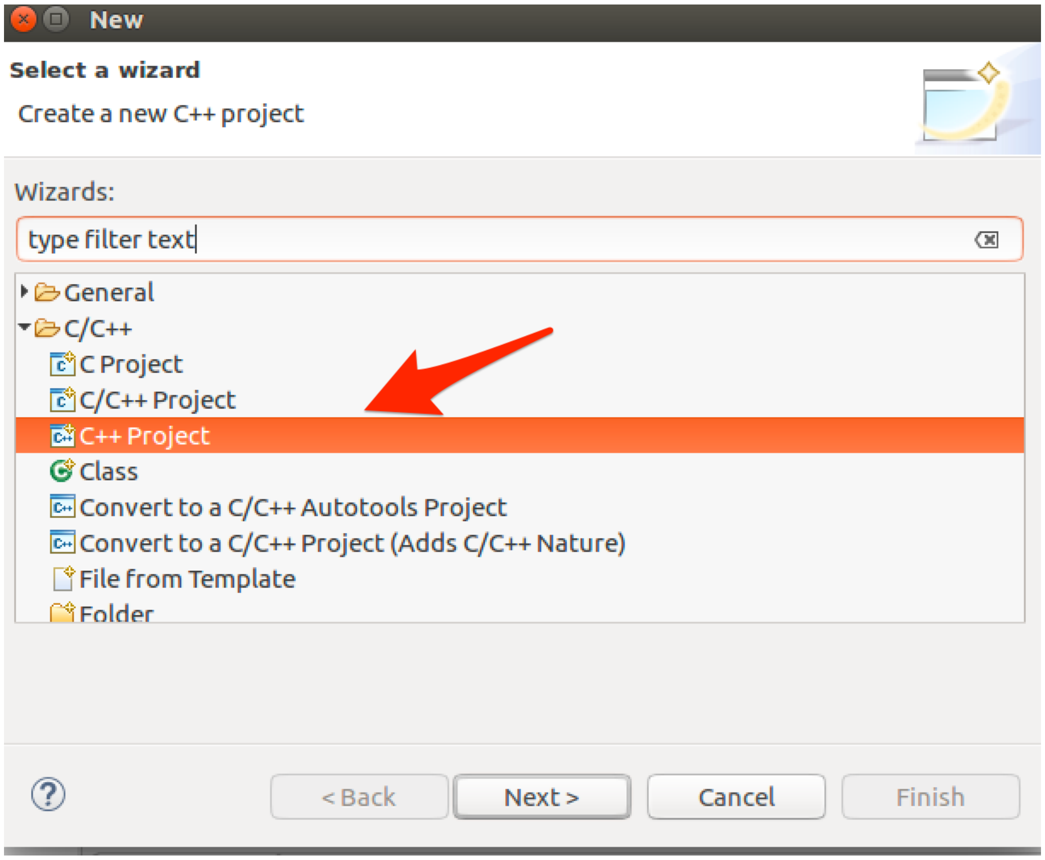
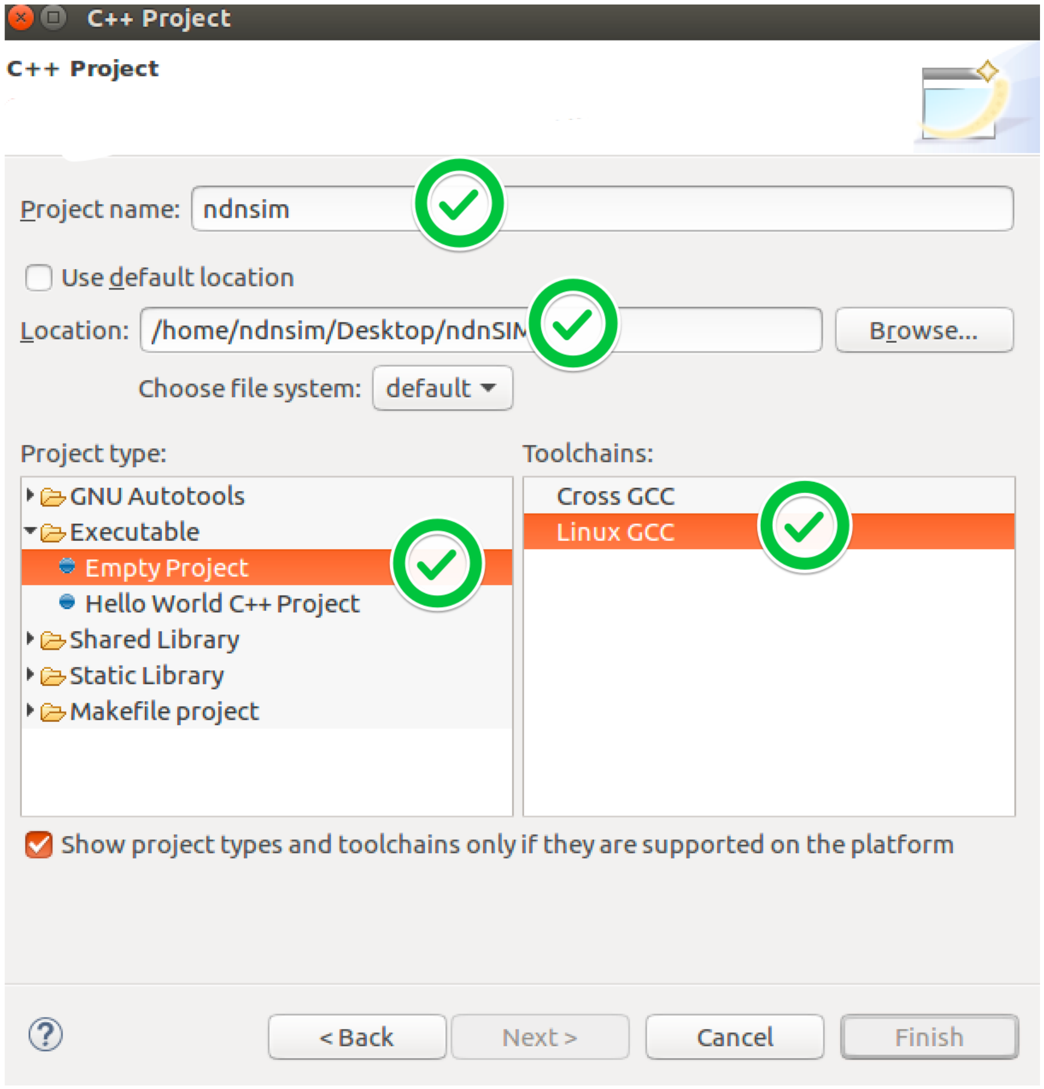


Figure 4: Define ndnSIM Directory in Eclipse

You need to careful defining the project name, location and executable Linuc GCC. After that you can finalize the steps with;



C++ Project

C++ Project

Project name: ndnsim

[ ] Use default location

Location: /home/ndnsim/Desktop/ndnSIM

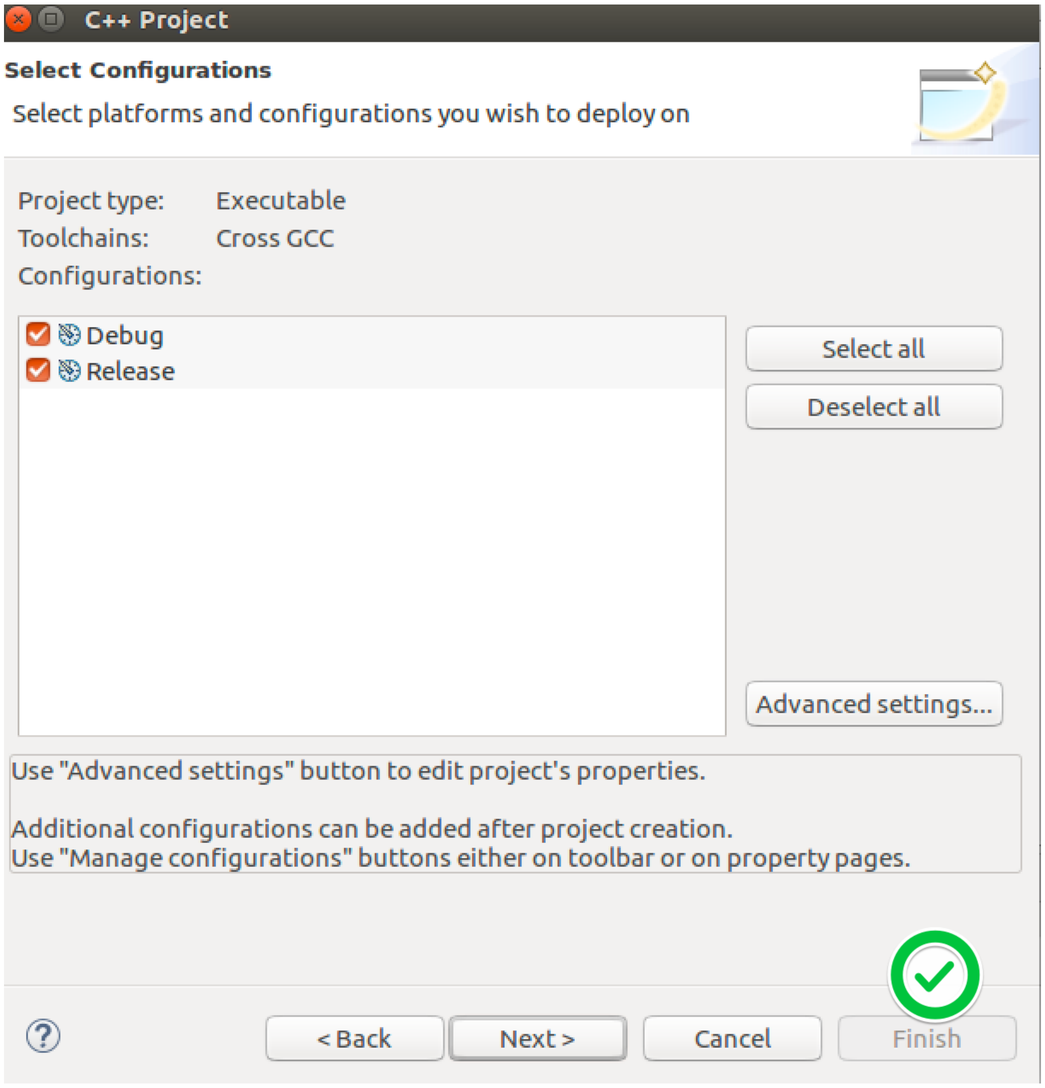
Choose file system: default

Project type:
- GNU Autotools
- Executable
  - Empty Project
  - Hello World C++ Project
- Shared Library
- Static Library
- Makefile project

Toolchains:
- Cross GCC
- Linux GCC

[✓] Show project types and toolchains only if they are supported on the platform

Figure 5: Extract ndnsim to Eclipse



Select Configurations

Select platforms and configurations you wish to deploy on

Project type:	Executable
Toolchains:	Cross GCC
Configurations:	

- ✓ Debug
- ✓ Release

Use "Advanced settings" button to edit project's properties.

Additional configurations can be added after project creation.
Use "Manage configurations" buttons either on toolbar or on property pages.

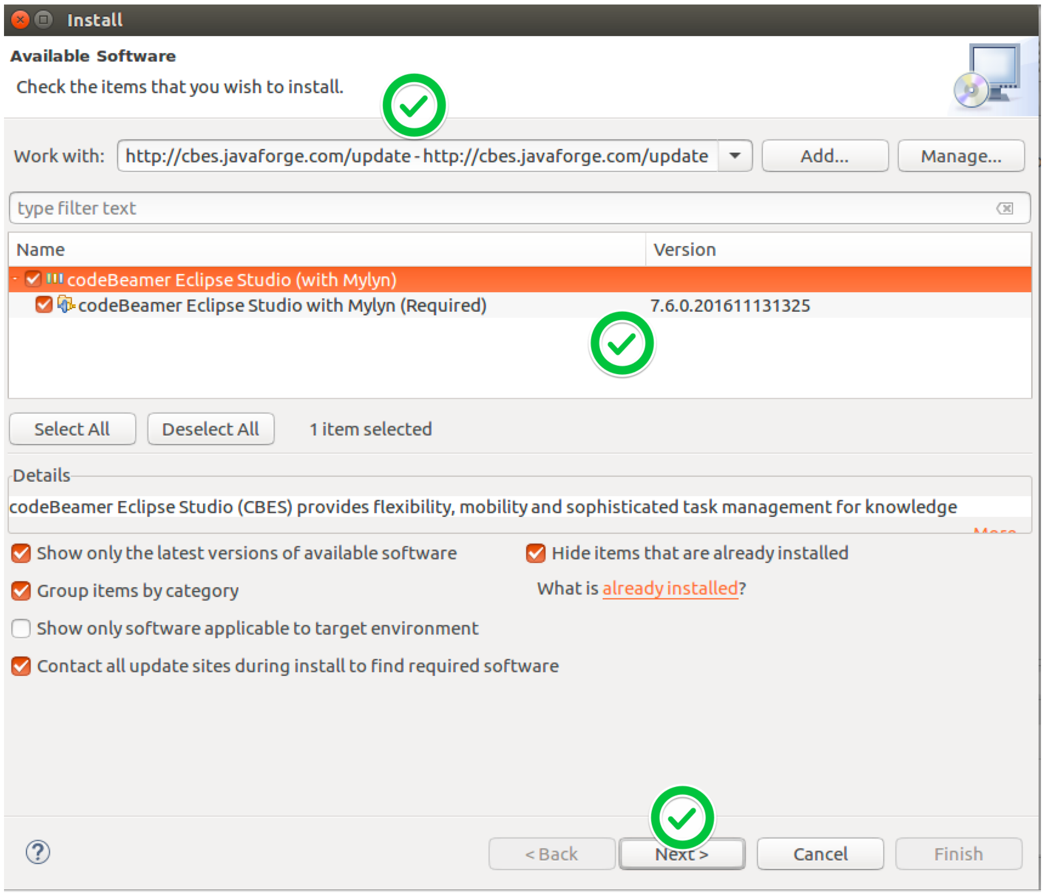
Figure 6: Extract ndnsim to Eclipse

5.1 Mercurial Setup

The Elipse requires Mercurial installation if you going to work on ndnSIM or ns-3. Firstly, you need to add a plug-in from **Help¿Install New Software**. After that, you need add a link

`http://cbes.javaforge.com/update`

After that, it will ask you to install mercurialeclipse plug-ins. Something like this;



Install

Available Software

Check the items that you wish to install.

Work with: http://cbes.javaforge.com/update - http://cbes.javaforge.com/update

type filter text

Name	Version
✓ codeBeamer Eclipse Studio (with Mylyn)	
✓ codeBeamer Eclipse Studio with Mylyn (Required)	7.6.0.201611131325

1 item selected

Details

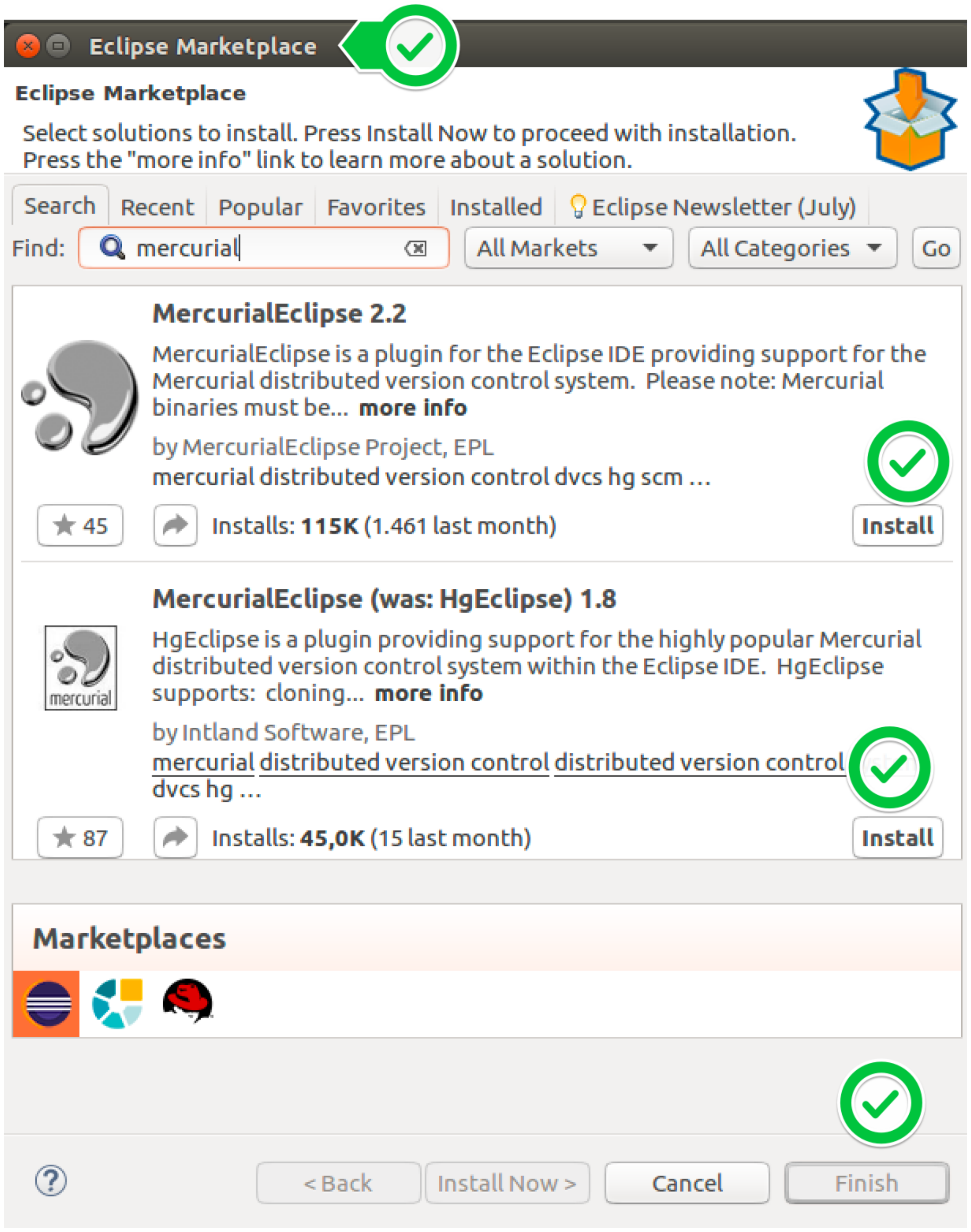
codeBeamer Eclipse Studio (CBES) provides flexibility, mobility and sophisticated task management for knowledge

- [✓] Show only the latest versions of available software
- [✓] Hide items that are already installed
- [✓] Group items by category
- [ ] Show only software applicable to target environment
- [✓] Contact all update sites during install to find required software

What is already installed?

Figure 7: Mercurial Set-up

If the beamers will not show you the Mercurial, you can go to `Help>Eclipse Market Place` then type Mercurial and follow the instructions.



Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

MercurialEclipse 2.2

MercurialEclipse is a plugin for the Eclipse IDE providing support for the Mercurial distributed version control system. Please note: Mercurial binaries must be... **more info**

by MercurialEclipse Project, EPL

mercurial distributed version control dvcs hg scm ...

★ 45 Installs: **115K** (1.461 last month) **Install**

MercurialEclipse (was: HgEclipse) 1.8

HgEclipse is a plugin providing support for the highly popular Mercurial distributed version control system within the Eclipse IDE. HgEclipse supports: cloning... **more info**

by Intland Software, EPL

mercurial distributed version control distributed version control dvcs hg ...

★ 87 Installs: **45,0K** (15 last month) **Install**

Marketplaces

Figure 8: Mercurial Set-up

5.2 Boost C++

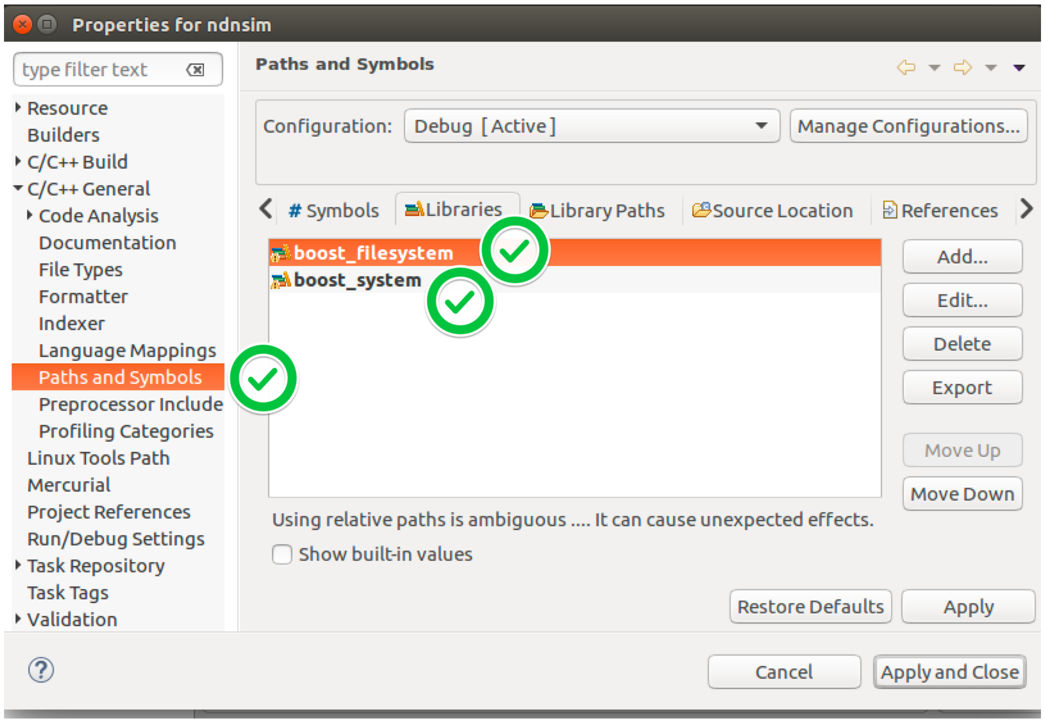
When you follow the instructions for installation ndnsim, you also install C++ Boost libraries as well. However, some projects/examples may require extra boost libraries that you have to define manually. You don't need to get the all related libs. but I'm going to define all of them to make the things easy and less problematic in your future work.

```
sudo apt-get install gcc g++ python python-dev mercurial bzr gdb
valgrind gsl-bin libgsl0-dev libgsl0ldbl flex bison tcpdump sqlite
sqlite3 libsqlite3-dev libxml2 libxml2-dev libgtk2.0-0 libgtk2.0-dev
uncrustify doxygen graphviz imagemagick texlive texlive-latex-extra
texlive-generic-extra texlive-generic-recommended texinfo dia texlive
texlive-latex-extra texlive-extra-utils texlive-generic-recommended
texi2html python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev
python-pygccxm
```

5.3 Boost Libs Definition on Eclipse

The eclipse can be not found Boost C++ libraries. To make this definition manually, you can follow these instructions. Right click to the project and go to properties and then manually `add` to the libraries;

```
boost_filesystem
boost_system
```



Configuration: Debug [Active]

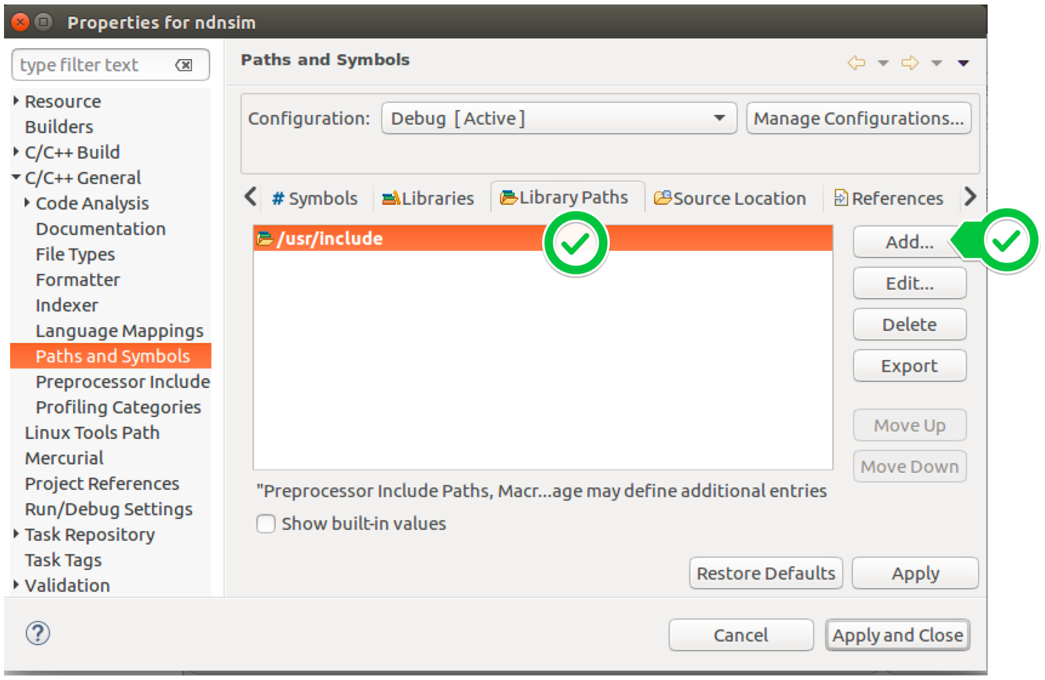
boost_filesystem
boost_system

Using relative paths is ambiguous .... It can cause unexpected effects.

Show built-in values

Figure 9: Boost Definition on Eclipse

After these definiton, you need to define Boost libs path. In my system, the Boost libs located on `/usr/include`



Configuration: Debug [Active]

/usr/include

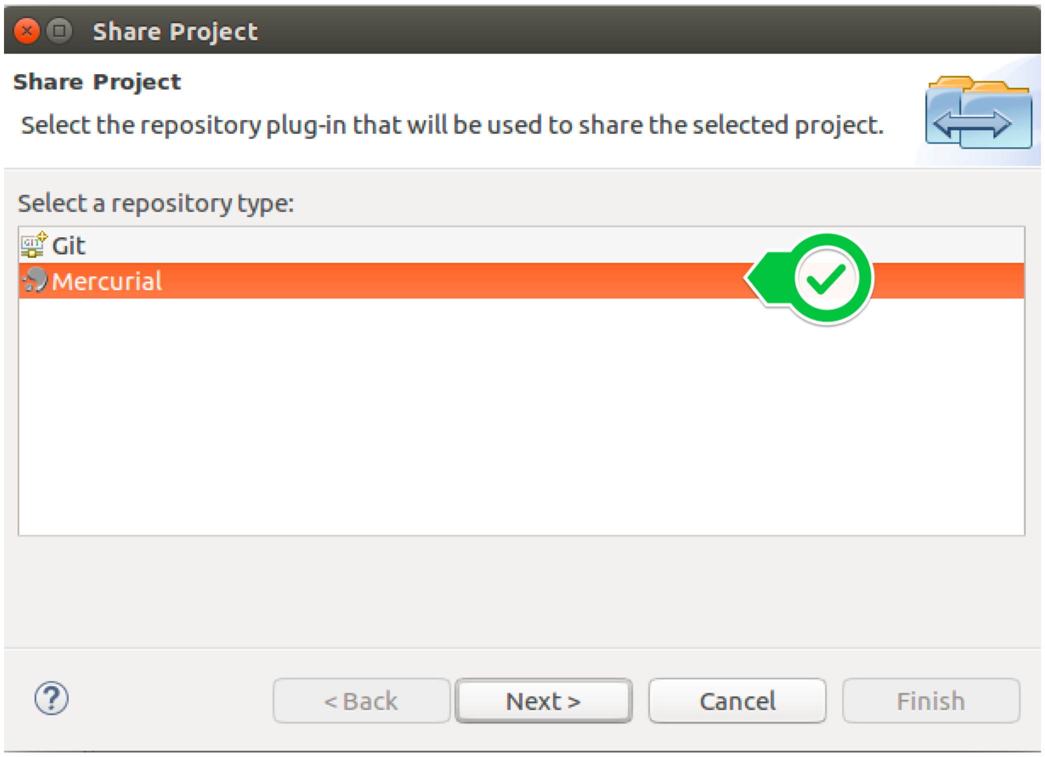
"Preprocessor Include Paths, Macr...age may define additional entries

Show built-in values

Figure 10: Boost Definition on Eclipse

5.4 Share Project

When the ndnsim directory has been indexed, you can define the mercurial for the ndnsim directory. You need to right click on project directory and then you need to select `Team>Share Project>Mercurial`. Finally, click on next and click on use local hg and finish.



Select the repository plug-in that will be used to share the selected project.

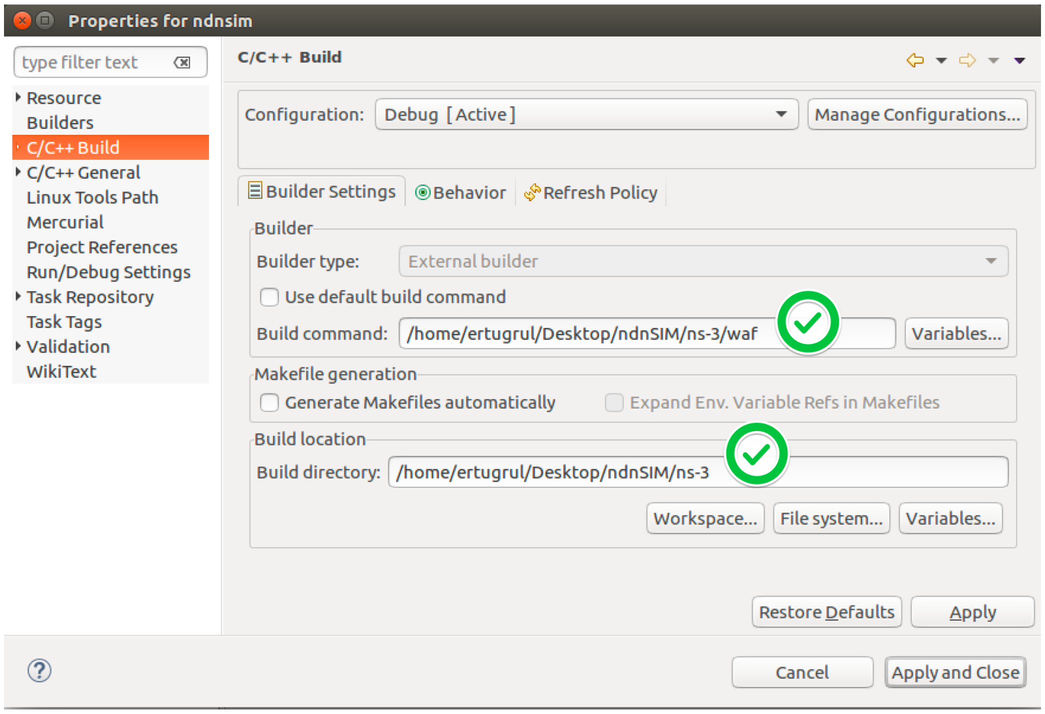
Select a repository type:

Git
Mercurial

Figure 11: Share Project

5.5 C++ Builder Setting

In this step, you need to define the builder wherever your ndnSIM directory. Here in this example, it defined at `/home/ertugrul/Desktop/ndnSIM/ns-3`. Do not forget to add `/waf` at build command with no ".".


Properties for ndnsim

C/C++ Build

Configuration: Debug [Active]

Builder Settings | Behavior | Refresh Policy

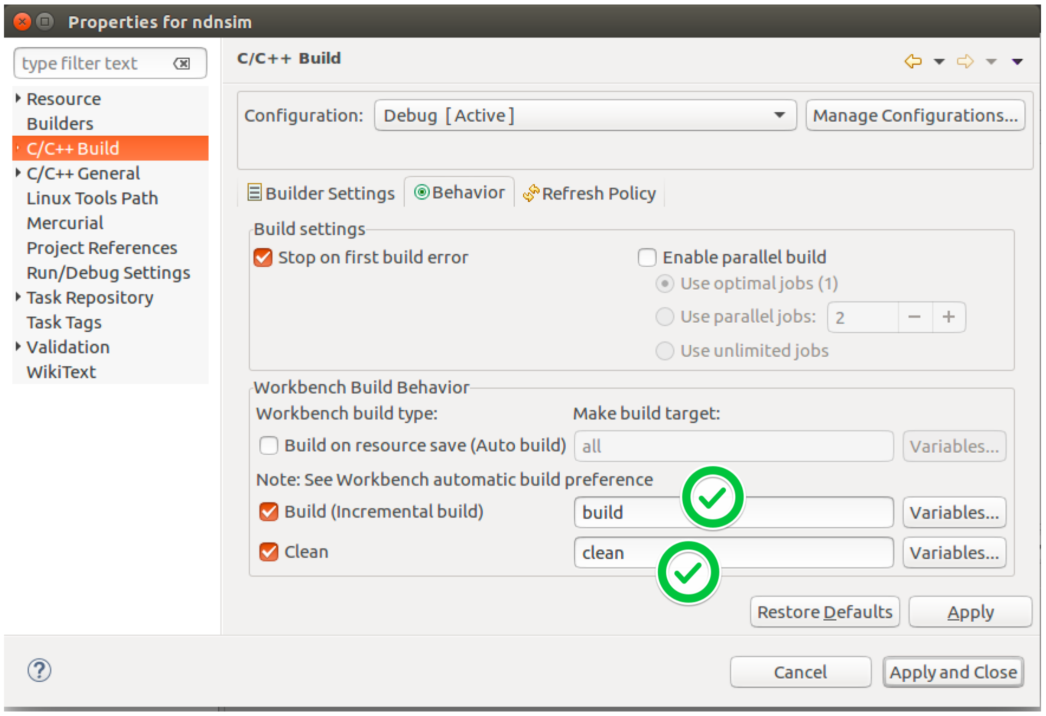
Builder
Builder type: External builder
[ ] Use default build command
Build command: /home/ertugrul/Desktop/ndnSIM/ns-3/waf

Makefile generation
[ ] Generate Makefiles automatically
[ ] Expand Env. Variable Refs in Makefiles

Build location
Build directory: /home/ertugrul/Desktop/ndnSIM/ns-3

Figure 12: Builder Setting

You also need to do the behaviour setting by followings ;



Configuration: Debug [Active]

Build settings
- [✓] Stop on first build error
- [ ] Enable parallel build
  - (●) Use optimal jobs (1)
  - ( ) Use parallel jobs: 2
  - ( ) Use unlimited jobs

Workbench Build Behavior

Workbench build type: Make build target:
- [ ] Build on resource save (Auto build) — all

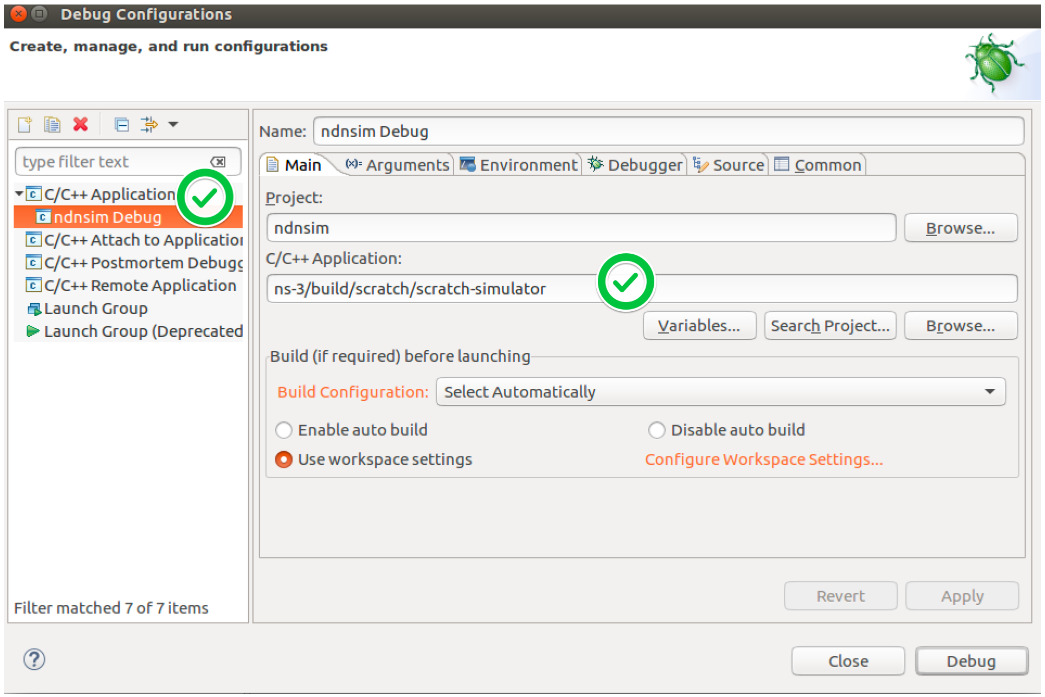
Note: See Workbench automatic build preference
- [✓] Build (Incremental build) — build
- [✓] Clean — clean

Figure 13: Behaviour Setting

5.6 Debug Configuration

You can also do debug by defining the ndnSIM application. Here in this example I defined scratch simulator.

Type under the Name field `"LD_LIBRARY_PATH"` (or `"DYLD_LIBRARY_PATH"` if you are on a MacOSX). Be sure the ”Append environment to native environment” is selected.



Name: ndnsim Debug

Project: ndnsim

C/C++ Application: ns-3/build/scratch/scratch-simulator

Build (if required) before launching

Build Configuration: Select Automatically

Enable auto build

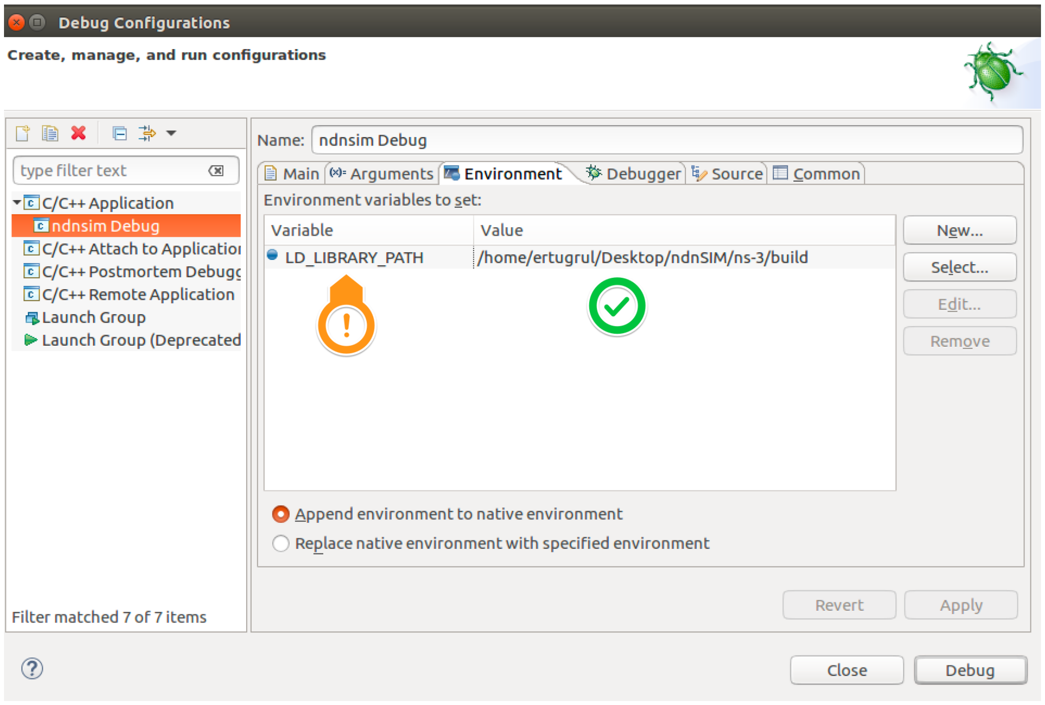
Disable auto build

[✓ Use workspace settings]

Figure 14: Debug Setting

5.7 Make Targets

Before being able to run waf for ndnSIM scenarios, we have to run ./waf configure. This can also be done within Eclipse. The way I have found to to this is we right click on the project and search for Make targets-¿Create:



Name: ndnsim Debug

Environment variables to set:

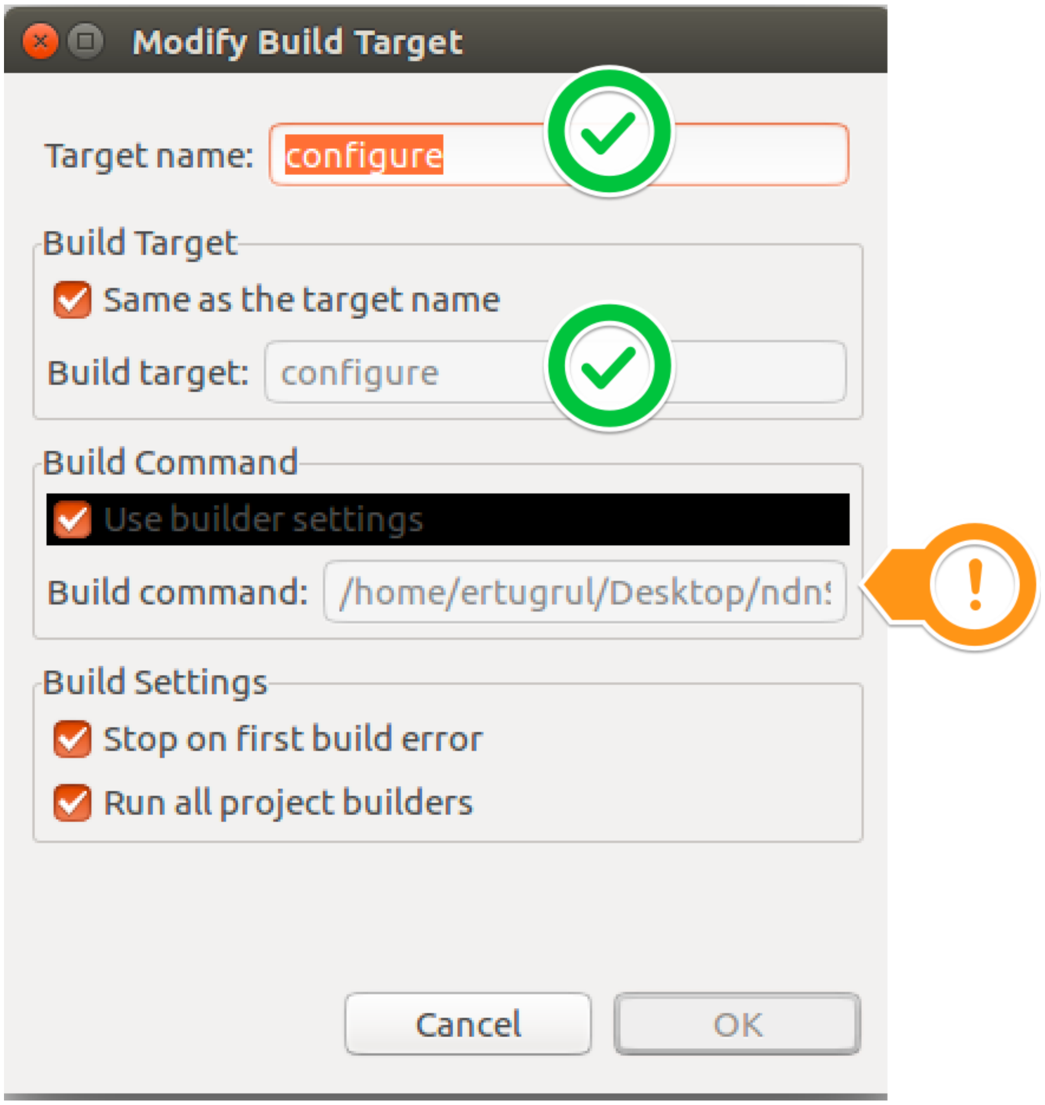
Variable	Value
LD_LIBRARY_PATH	/home/ertugrul/Desktop/ndnSIM/ns-3/build

[✓ Append environment to native environment]
Replace native environment with specified environment

Figure 15: Debug Setting

6 CDT Build Console

Finally, when you can Build Project by right clicking. When you succeeded all steps so far, you need see something like this output on Eclipse's console.



Modify Build Target

Target name: configure

Build Target

✓ Same as the target name

Build target: configure

Build Command

✓ Use builder settings

Build command: /home/ertugrul/Desktop/ndn!

Build Settings

✓ Stop on first build error

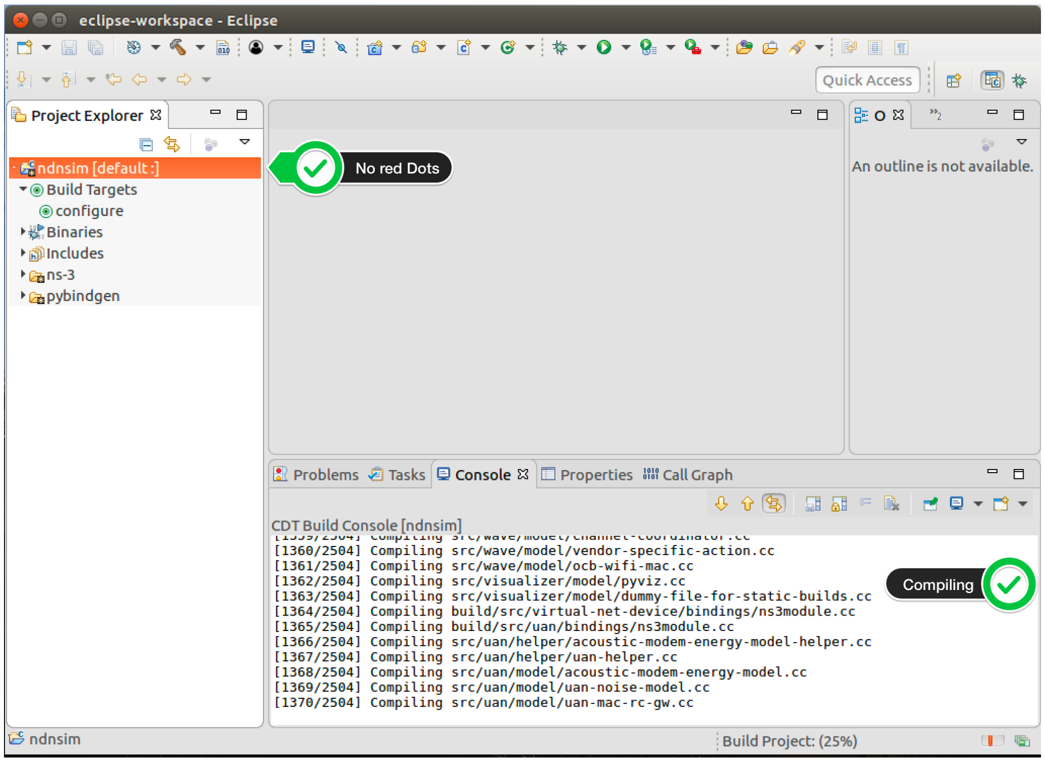
✓ Run all project builders

Figure 16: Make Targets

7 Location of your Own Scenario

Simulation scenarios can be written directly inside NS-3 in `scratch/` or `src/ndnSIM/examples` folder.

Alternative and a recommended way is to write simulation scenarios in a separate repository, not related to either NS-3 or ndnSIM. For example, you



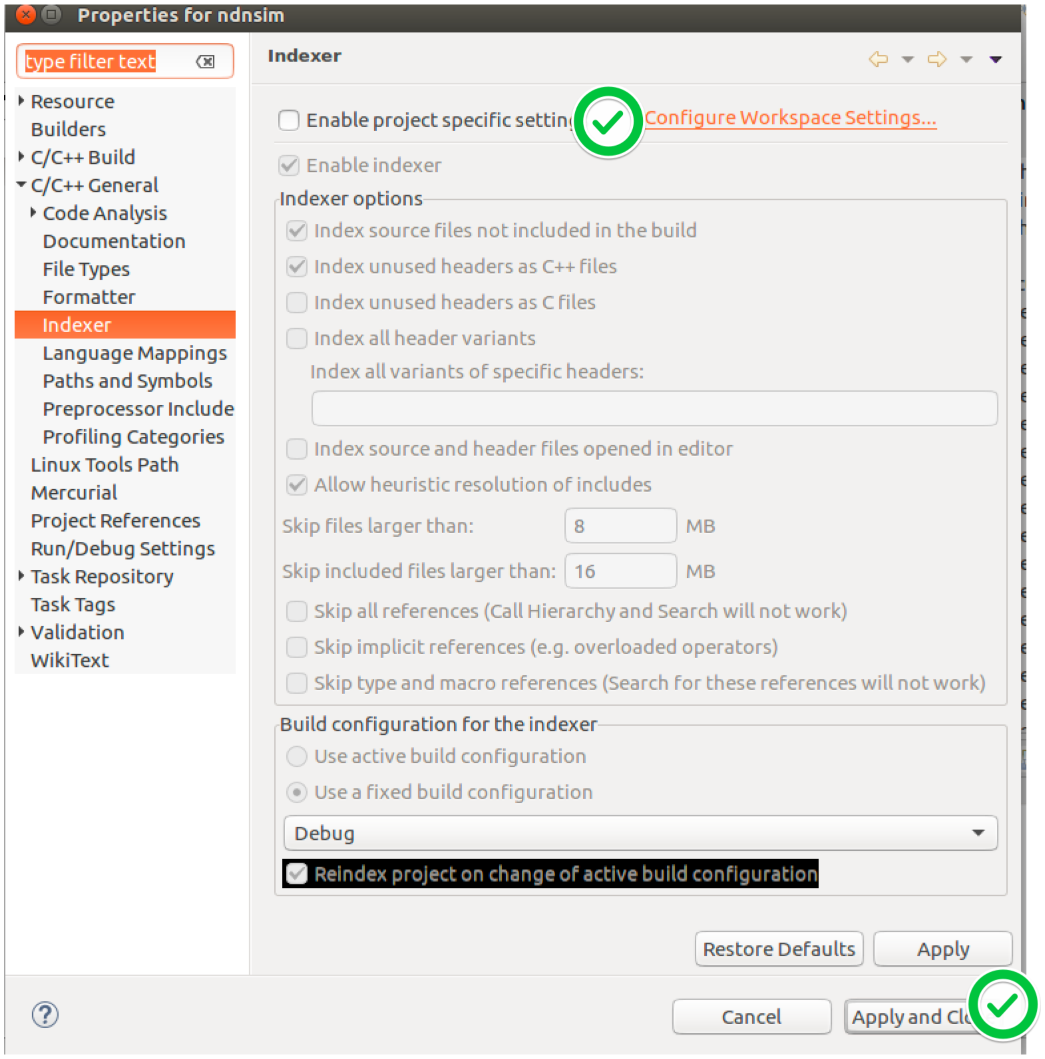
CDT Build Console [ndnsim]

```
[1360/2504] Compiling src/wave/model/vendor-specific-action.cc
[1361/2504] Compiling src/wave/model/ocb-wifi-mac.cc
[1362/2504] Compiling src/visualizer/model/pyviz.cc
[1363/2504] Compiling src/visualizer/model/dummy-file-for-static-builds.cc
[1364/2504] Compiling build/src/virtual-net-device/bindings/ns3module.cc
[1365/2504] Compiling build/src/uan/bindings/ns3module.cc
[1366/2504] Compiling src/uan/helper/acoustic-modem-energy-model-helper.cc
[1367/2504] Compiling src/uan/helper/uan-helper.cc
[1368/2504] Compiling src/uan/model/acoustic-modem-energy-model.cc
[1369/2504] Compiling src/uan/model/uan-noise-model.cc
[1370/2504] Compiling src/uan/model/uan-mac-rc-gw.cc
```

Figure 17: Console Output

can use the following template to write your extensions, simulation scenarios, and metric processing scripts: ndnSIM scenario Template.

8 Indexer and C/C++ Settings



Properties for ndnsim

Indexer

- [ ] Enable project specific settings — Configure Workspace Settings...
- [✓] Enable indexer

Indexer options
- [✓] Index source files not included in the build
- [✓] Index unused headers as C++ files
- [ ] Index unused headers as C files
- [ ] Index all header variants

Index all variants of specific headers:

- [ ] Index source and header files opened in editor
- [✓] Allow heuristic resolution of includes

Skip files larger than: 8 MB

Skip included files larger than: 16 MB

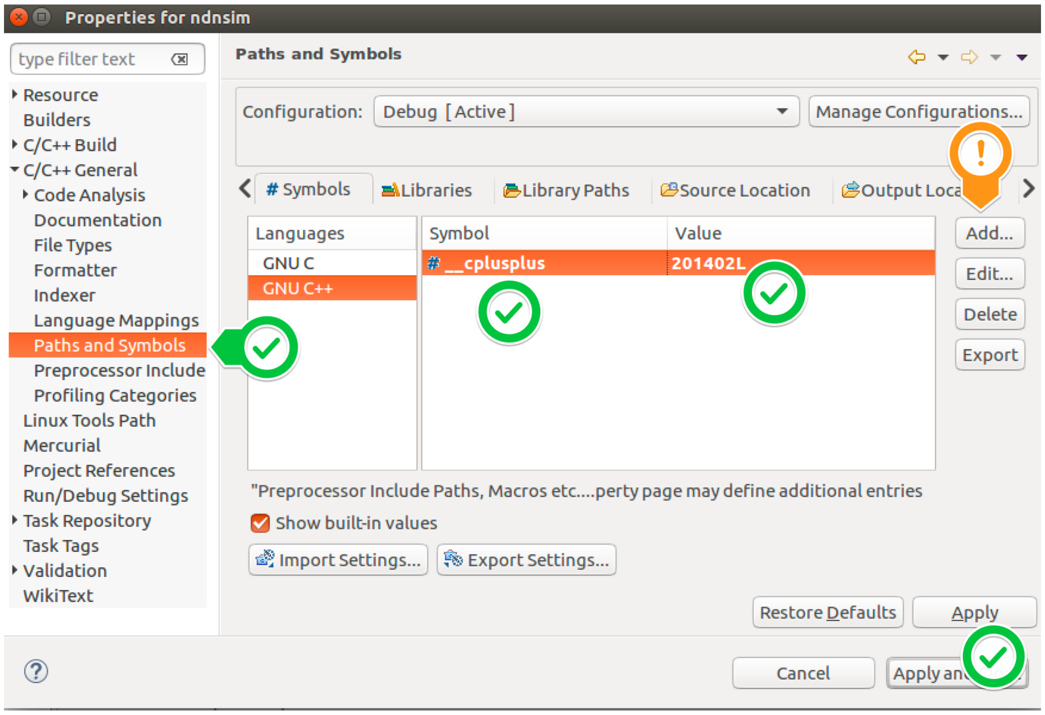
- [ ] Skip all references (Call Hierarchy and Search will not work)
- [ ] Skip implicit references (e.g. overloaded operators)
- [ ] Skip type and macro references (Search for these references will not work)

Build configuration for the indexer
- ○ Use active build configuration
- ● Use a fixed build configuration

Debug

- [✓] Reindex project on change of active build configuration

Figure 18: Indexer Setting



Paths and Symbols

Configuration: Debug [Active]

Languages	Symbol	Value
GNU C		
GNU C++	# __cplusplus	201402L

"Preprocessor Include Paths, Macros etc....perty page may define additional entries

☑ Show built-in values

Figure 19: Path Setting